MASTER EN SCIENCES INFORMATIQUES

Outil d'aide à l'évaluation des pratiques logicielles suivant le modèle OWPL

Makunda Sefekese, Bobo

Award date:
2001

Facultés Universitaires Notre-Dame de la Paix Namur
Institut d'informatique
Rue Grandgagnage 21
B-5000 Namur

Année académique 2000-2001

Outil d'aide à l'évaluation des pratiques logicielles suivant le modèle OWPL

Bobo MAKUNDA SEFEKESE

Mémoire présenté en vue de l'obtention du grade de Maître en informatique

Résumé

Le développement des logiciels est un processus qui a ses caractéristiques et qui doit être évalué pour procéder à son amélioration.
Plusieurs méthodes sont mises en œuvre pour faire ces améliorations, notamment les évaluations suivant les modèle SPICE et CMM. Ces modèles sont trop complexes et trop détaillés pour qu'ils soient appliqués aux PME wallonnes. Les PME Wallonnes sont souvent de petites équipes, avec des étapes très particulières dans le développement des logiciels. C'est pourquoi le modèle OWPL (*Observatoire Wallon des Pratiques Logicielles*) a été créé pour pouvoir évaluer et améliorer les pratiques logicielles dans les PME Wallonnes.
Le but de ce mémoire est de concevoir un outil qui gère le modèle OWPL. Cet outil doit gérer tous les documents d'évaluation (les questionnaires, le glossaire, la description du modèle, et les facteurs de succès). Une exigence supplémentaire à cet outil est qu'il doit être portable, car il sera utilisé dans les entreprises qui travaillent sur des plates-formes différentes.

Abstract

Developing software is a process which has its own characteristics and must be evaluated for improvement to take place. Several methods are implemented to make these improvements like evaluations according to the SPICE and the CMM models. These models are too complex and too detailed to be apply with small Walloon businesses. These small businesses are often small teams, applying very particular steps in the development of the software.
This is why the OWPL (*Observatoire Wallon des Pratiques Logicielles*) model was created to be able to evaluate the software practices in small Walloon businesses.
The goal of this thesis is to design a tool which manages the model. This tool must manage all the documents used for evaluation according to the OWPL model. An additional requirement for this tool is that it must be multi-platform, as it will be used in companies which work on various platforms.

Avant toute chose, j'aimerai exprimer toute ma gratitude à l'égard du professeur Naji HABRA et à Alain RENAULT, sans qui le présent travail n'aurait pu être réalisé. La qualité de leur enseignement, leur soutien et leur encadrement m'ont été profitables aussi bien lors du déroulement de mon stage à Charleroi que lors de la rédaction de mon mémoire. Qu'ils trouvent à travers ces mots, toute ma reconnaissance envers eux.

Il est des êtres qui me sont chers, des êtres sans qui je ne serais pas sur cette terre: mes parents. Sans leur affection, leur soutien moral, matériel et financier, je n'aurai pu arriver à mes fins. Les mots ne pourront jamais traduire toute ma gratitude à leur égard, mais je voudrais tout de même qu'ils sachent que je ferai tout ce qui est en mon pouvoir pour être à la hauteur de la confiance qu'ils placent en moi.

Papa, ya Catho et maman, je vous remercie pour tout ce que vous ne cessez de faire pour moi. Ma gratitude s'adresse également à mes chères sœurs, à mon cher frère, ainsi qu'à tous les membres de notre grande famille.

Mon plus grand vœux aurait été que certaines personnes partagent ma joie, mais le destin en a décidé autrement. J'ai ici une pensée pour ma sœur, mes grands-mères et mes tantes et oncle qui nous ont quittés il y a quelques années. Ces personnes ont toutes contribué, de près ou de loin, consciemment ou inconsciemment, à la réussite de mes études.

Je voudrais aussi remercier tous mes amis, la communauté des étudiants Congolais de Namur, les habitants du "75" à Namur, le révérend père DEVOS et feu le révérend père GRIFFE, pour avoir contribué à un environnement favorable au déroulement de mes études. La liste serait longue si je devais citer les noms de tous mes amis qui me sont chers, mais j'aimerais tout spécialement remercier Jimmy BATILA et Stéphanie MUNDERE pour leur soutien efficace à tous mes projets.

Je m'en voudrais de terminer cette liste de remerciements sans exprimer toute ma reconnaissance à l'égard des personnes qui nous ont encadrés aux Facultés Universitaires Notre-dame de la Paix. Toute ma gratitude à tous mes professeurs et assistants, à la commission "Etudiants Etrangers", à monsieur HONOREZ du secteur social, et à toutes les personnes qui ont contribué de manière indirecte pour que je puisse étudier dans un environnement très enrichissant tant sur le plan humain qu'académique. Mes remerciements particuliers à Aimé KASSA et Efrem MBAKI pour leur assistance au delà des horaires légaux.

Enfin, et non les moindres, je voudrais remercier tous mes camarades de promotion. Il y a ceux avec qui nous avons fait du chemin pendant cinq ans et d'autres moins. Je voudrais tout simplement les remercier pour tous les bons moments que nous avons passés ensemble tout au long de nos années fac.

TABLE DES MATIERES

I. INTRODUCTION GENERALE

Introduction générale

Aucune entreprise ne peut prétendre à une productivité de qualité constante à moyen ou à long terme si elle n'investit dans une démarche qualité. Certaines le font réellement pour améliorer et standardiser leurs pratiques internes (relation entre collaborateurs) et externes (relation avec les clients). Dans certains cas, cette démarche qualité conduit à une certification qui garantit l'image de marque de l'entreprise.

Dans la foulée des différentes démarches qualité, la qualité de développement des logiciels est vite devenue la pierre angulaire des développeurs dès qu'ils se sont rendu compte que le développement des logiciels est en soi une activité qui mérite rigueur et organisation. Désormais l'assurance d'une réussite des projets informatiques ne réside pas uniquement dans la compétence des employés, mais aussi et surtout dans toute l'organisation autour du développement des logiciels.

C'est dans ce cadre que plusieurs modèles d'évaluation ont vu le jour. Les modèles CMM et SPICE se sont vite imposés comme standards, et leur utilisation est aujourd'hui universelle chez les consultants de qualité logicielle.

Bien que ces modèles se soient imposés, le coût élevé de recours à des entreprises de qualité logicielle qui les utilisent et leur degré de complexité constituent un frein pour de petites structures organisationnelles comme les PME Wallonnes. Une adaptation de ces modèles aux spécificités des PME Wallonnes était plus que souhaitable: cette adaptation a vu le jour avec le projet Observatoire Wallon des Pratiques Logicielles, OWPL en sigle.

Le projet OWPL a mis en place un modèle d'évaluation des pratiques logicielles dans les PME Wallonnes. Afin de ne pas partir de rien, et de ne pas adopter une démarche isolée, le modèle OWPL s'est largement inspiré des modèles CMM et SPICE pour offrir un modèle efficace adapté à toute démarche qualité des PME Wallonnes sur le développement des logiciels. D'ores et déjà, le modèle OWPL s'est plus investi dans l'aspect amélioration des pratiques logicielles que dans la certification d'un label de qualité.

Appliquer le modèle dans une PME Wallonne nécessite parfois une adaptation du modèle OWPL à cette entreprise, en vue d'aplanir les écarts éventuels pouvant exister entre la nomenclature des notions utilisées dans le modèle OWPL et celle utilisée dans la PME. Cette adaptation passe par une révision manuelle de tous les documents décrivant le modèle OWPL.

L'évaluation suivant le modèle OWPL se fait à l'aide des questionnaires à remplir lors des interviews avec les employés d'une PME. Le remplissage des formulaires, et des grilles des cotations sont aussi fait manuellement.

Le but du présent travail est de proposer une application de gestion automatique du modèle OWPL. Cette application doit permettre de gérer tous les outils du modèle OWPL utilisés lors de l'évaluation d'une entreprise donnée. Lors de l'utilisation de cette application, les relations existantes entre les différents outils doit persister, et l'application doit pouvoir être utilisée de manière transparente sur des plates-formes différentes. Cette exigence est facile à comprendre dans le mesure où les entreprises évaluées utilisent chacune un environnement de travail particulier: Il est donc déterminant que l'application de gestion du modèle OWPL soit portable.

C'est dans ce cadre que nous avons effectué un stage d'octobre 2000 à janvier 2001 à l'Observatoire Wallon des Pratiques Logicielles qui se trouve dans les enceintes de l'antenne de Charleroi des Facultés Universitaires Notre-Dame de la Paix.

Le présent travail fait état de toute la démarche qui a été menée pour aboutir à une application de gestion du modèle OWPL qui soit portable.

Pour aboutir à cette application, deux choix techniques ont été nécessaires : trouver un gestionnaire de base des données et utiliser une technologie de développement portables. Nous présenterons les différentes possibilités envisagées, puis nous justifierons nos choix suivant les critères de départ : d'une part, trouver un excellent système de gestion des bases des données n'est pas une opération difficile en soi, mais exiger en plus la portabilité est plus que délicat. D'autre part, plusieurs langages de programmation offrent largement la possibilité d'implémenter un système de gestion d'un modèle tel que le modèle OWPL, mais, malheureusement, peu d'entre eux offrent une portabilité de manière la plus transparente possible pour l'utilisateur.

Le contenu du présent mémoire commence par une présentation du modèle OWPL. Après cette présentation, nous verrons ce que nous avons proposé comme modélisation du modèle OWPL à partir des différents documents qui le constitue. Après une prise de connaissance des exigences de l'utilisateur de l'application, nous réfléchirons sur les différentes technologies possibles pour implémenter un système comme celui qui nous est demandé. Nous verrons ensuite l'architecture que nous proposons pour notre application sous forme de découpe en sous-tâches et du diagramme des différents flux d'informations.

Nous procéderons à une présentation de la partie que nous avons pu implémenter. Dans ce prototype, nous avons tenu à implémenter une partie aussi autonome et complète que possible dans l'utilisation de toutes les technologies utilisées, en vue de procéder dans un premier temps à un ensemble des tests de l'utilisateur. Cette manière de procéder nous a ainsi permis d'apporter quelques précisions supplémentaires concernant la volonté et les attentes de l'utilisateur. C'est donc uniquement ce prototype qui est présenté ici: il n'est pas suffisamment complet pour être déployé en vue d'une utilisation grandeur nature, mais il permet à l'utilisateur de faire quelques tests sur le modèle réel et constitue un outil qui est très proche de l'application finale. Nous terminerons notre travail avec une conclusion générale sur le sujet abordé.

II. DOMAINE D'APPLICATION

CHAPITRE 1. Le modèle OWPL

"L'objectif du projet OWPL (Observatoire Wallon des Pratiques logicielles) est de définir un modèle d'évaluation et d'amélioration des processus de production de logiciels adapté aux petites structures dans le but d'aider ces petites structures à réellement améliorer leurs pratiques logicielles"[1].

A l'instar des modèles SPICE (*Software Process Improvement and Capability dEtermination*) et CMM (*Capability Maturity Model*) , le modèle OWPL a été créé dans le cadre du projet de même nom pour évaluer et surtout améliorer les processus logiciels dans les PME Wallonnes. Un tel projet spécifique aux PME Wallonnes trouve son origine dans le fait que les modèles SPICE et CMM s'avèrent trop complexes, trop coûteux, et non adaptés aux spécificités des PME Wallonnes, généralement de taille réduite, avec une structure peu complexe, un nombre limité d'acteurs et un niveau modeste de maturité de processus.
Selon une étude réalisée par la CITA[1], le recours à une société spécialisée dans l'amélioration des pratiques logicielles peut représenter 50% du budget annuel de près de 42% des PME Wallonnes[2]. Le modèle OWPL permet aux PME Wallonnes d'opter pour une démarche qualité à moindre coût avec une procédure moins lourde et moins complexe.

Le modèle OWPL est plus adapté à ces entreprises et permet de mettre l'accent sur leurs points forts et leurs points faibles, en vue de l'amélioration de leurs processus logiciels, et ce suivant une démarche progressive et guidée.

Avant de passer à une description du modèle OWPL, voyons d'abord ce que sont les modèles SPICE et CMM.

1.1. Le modèle SPICE (*Software Process Improvement and Capability dEtermination*)

SPICE est le nom du projet à l'origine du standard 15504 de ISO[2]/IEC[3]. A l'origine, le projet voulait aboutir à une méthode et un modèle d'évaluation des processus logiciels, mais le résultat final du projet (le standard ISO/IEC 15504) propose un ensemble d'exigences relatives à des méthodes et des modèles conformes à cette norme. Au final, même s'il est abusif de parler du modèle SPICE, le standard ISO/IEC 15504 propose dans un de ses documents (ISO/IEC contient 9 documents, dont 3 documents normatifs et 6 documents descriptifs) un modèle qui est conforme à la même norme, et c'est ce modèle qu'on peut appeler le modèle SPICE.

L'évaluation suivant ce modèle s'effectue donc par processus, les processus étant regroupés par catégories : processus relation clients-fournisseurs, processus d'ingénierie, processus de support, processus de gestion et le processus d'organisation.
A terme, dès 2001, ISO/IEC 15504 deviendra le standard auquel une méthode ou un modèle doit se conformer. C'est notamment une démarche qui est prévue pour le modèle CMM dont nous parlons dans le prochain point.

---

[1] Cellule Interfacultaire de Technologie Assessment, FUNDP Namur.
[2] International standardisation Organisation : organisme international des standards internationaux avec plus de 9200 séries de normes dont les séries ISO9000 qu définissent les exigences en terme de système qualité en fonction de type de l'activité(des activités).
[3] International Eletronic Commission

1.2. Le modèle CMM (*Capability Maturity Model*)

Le modèle CMM existe depuis plus de 10 ans. La version 1.1 doit être bientôt remplacée par la version 2.0. Le modèle CMM est un référentiel des pratiques clés que doit mettre en œuvre toute entreprise pour le développement ou la maintenance de ses logiciels. Publié par le SEI[4], il est utilisé pour évaluer le niveau de maturité des processus logiciels des entreprises et organismes dans le cadre d'une démarche d'amélioration de qualité logicielle.

Dans le modèle CMM, chaque niveau de maturité contient des processus-clés et chaque processus-clé est composé d'un certain nombre de pratiques.
Comme exemple, on peut citer quelques processus du niveau 1-2 : gestion des exigences, gestion de projet, gestion des sous-contractants, assurance de la qualité, et gestion de la configuration. Les processus "coordination inter-groupes" et "revues par des paires" sont des exemples des processus du niveau 2-3.

1.3. Description du modèle OWPL

La première hypothèse du modèle OWPL est que toute activité d'une entreprise doit être effectuée dans l'optique de la réalisation des objectifs de l'entreprise: "*Chaque activité aura son objectif propre et concourra à la réalisation de l'objectif du processus auquel elle appartient. Elle apporte ainsi sa contribution à la réalisation de l'objectif du processus qui est lui-même défini en fonction de l'objectif global du département (Ex. : les processus logiciels). Cet objectif est à son tour défini eu égard à l'objectif général de l'entreprise. Les objectifs organisés de cette manière sont donc toujours en accord avec les objectifs de l'entreprise.*"[1]. La figure 1 ci-dessous présente da manière schématique l'arbre des objectifs d'une entreprise.

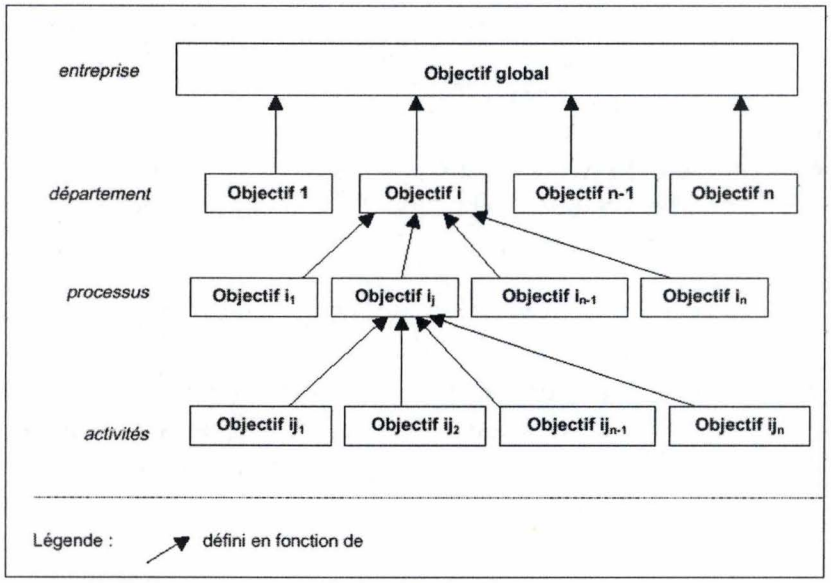


Figure 1: l'arbre des objectifs [1]

[4] Software Engineering Institute (http://www.sei.cmu.edu/cmmi/)

L'élément central de ce modèle est le **processus.** Un processus est défini en fonction d'un objectif qui contribue à la réalisation de l'objectif global de l'entreprise. Chaque processus est composé des **pratiques** nécessaires pour sa mise en œuvre effective. Le processus se déroule dans un environnement où des **facteurs de succès** garantissent sa performance. Un **glossaire** est utilisé pour décrire les termes techniques utilisés dans le modèle. Les liens entre ces différents éléments sont illustrées par la figure 2.

Les différents composant du modèle OWPL seront définis de manière plus détaillée au Chapitre 3 lors de la modélisation des objets du système à réaliser. Pour une connaissance détaillée du modèle OWPL au moment de la rédaction du présent document, vous pouvez vous référer au document numéro 1 en annexe.

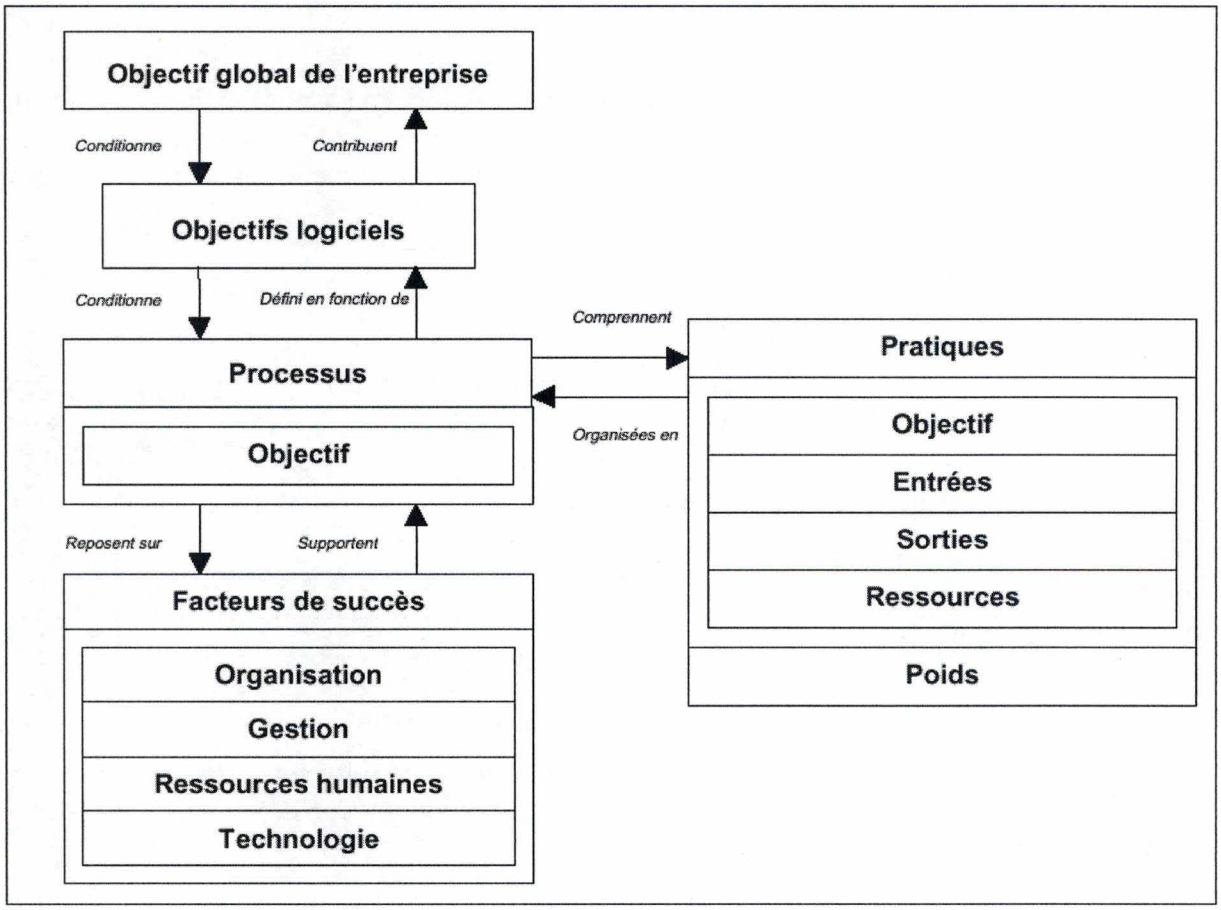


Figure 2: Structure du modèle OWPL[1]

Le modèle OWPL étant plus axé sur l'amélioration des processus dans une entreprise, une micro-évaluation est effectuée au préalable, avec toujours un questionnaire, en vue de révéler les forces et les faiblesses de l'entreprise. C'est cette micro-évaluation qui permet de déceler les processus qui vont être évalués.

1.4. Cotation suivant le modèle OWPL

L'évaluation suivant le modèle OWPL est basée sur un questionnaire composé de questions ouvertes portant d'une part sur les processus à évaluer, et d'autre part sur tous les facteurs de succès. Les réponses à ces questions sont évaluées en fonction d'une grille d'analyse systématique.

Pour faciliter l'interprétation des réponses des personnes interrogées lors des interviews, une méthode de cotation des questions est proposée avec le modèle OWPL. Cette méthode de cotation s'applique aux questionnaires sur les processus et aux questionnaires sur les facteurs de succès.

Pour les deux points suivants (Cotation des pratiques et Cotation des facteurs de succès), **les textes sont tirés tel quel de [6]**. On y trouve une explication de la cotation sur les questionnaires concernant les pratiques et les questionnaires concernant les facteurs de succès.

1.4.1 Cotation des pratiques

Intitulé de la question		**PROCESSUS : PROC** **PRATIQUE : PROC/PR01/02** **QUESTION : PROC-Q01**		
Sous-question susceptible d'orienter la personne interrogée		Pas applicable		
PROC-Q01-P1	Proposition 1	0	0	[Σ]
PROC-Q01-P2	Proposition 2	3	5	
PROC-Q01-P3	Proposition 3	7	10	
PROC-Q01-P4	Proposition 4	15	30	

La grille de cotation est une grille de **n lignes * 3 colonnes**. Les n lignes correspondent aux n propositions de réponses et les 2 premières colonnes correspondent au degré d'institutionnalisation de la pratique analysée.

En général :
Plusieurs cases peuvent être cochées dans la première colonne indiquant différents niveaux de qualité pour cette pratique en fonction des projets. Si une seule méthode est utilisée, sur tous les projets, on cochera uniquement la case correspondante dans le colonne 2. Chaque réponse est ainsi évaluée eu égard à la qualité de la pratique et à son degré d'institutionnalisation.

Cas particulier :
Si la troisième colonne contient le symbole $\sum$, cela signifie que les options proposées sont complémentaires et non exclusives. Ainsi elles peuvent être toutes cochées simultanément dans la colonne c1 ou dans la colonne c2 suivant qu'elles sont d'application sur tous les projets ou non.

La colonne C3 contient le report, c'est-à-dire la note globale à attribuer à la question en fonction des réponses obtenues. Si la troisième colonne contient le symbole $\sum$, le report est

égal à la somme des cotes correspondant aux différents propositions sélectionnées. Dans le cas contraire, il est le MAXIMUM de ces valeurs.

Soit la partie à cocher dans la grille de cotation représentée par la grille suivante :

	c1	c2	C3
L0		n.a.	Report
L1	U	X	
L2	V	Y	
L3	W	Z	

Règle de cotation:

Si la question est hors contexte, cochez dans (c2, l0)
Sinon : Cocher dans c1 **ou** c2
Si la c3 ∋ '∑'
C1 : cocher U et/ou V et/ou W
C2 : cocher X et/ou Y et/ou Z
Report = ∑ (valeurs des cases cochées)
Sinon
C1 : cocher U et/ou V et/ou W
C2 : cocher X ou Y ou Z
Report = MAX (valeurs des cases cochées)
∑ (U,V,W) < Z
∑ (reports) ▶ 90

Toute question jugée «**non applicable**» sera extraite du calcul, et le total des reports sera ramené sur le total maximum de la pratique concernée grâce à une règle de trois.

Les points attribués aux différentes propositions de réponse d'une question mettent en évidence la gradation qui existe entre le niveau de qualité de ces différentes propositions. De même, si plusieurs pratiques différentes sont parfois rencontrées pour une question, on ne tiendra compte que de la cote attribuée à la pratique de niveau le plus élevé afin de ne pas dépasser la cote correspondant à la meilleure pratique institutionnalisée.

Les différentes cotes attribuées aux propositions de réponse ne sont pas reprises sur le questionnaire afin d'éviter d'influencer l'évaluateur lorsqu'il est amené à sélectionner la (les) proposition(s) correspondant le mieux à la réponse effectivement produite par le témoin. Ces valeurs sont stockées à même l'outil d'analyse qui est utilisé pour produire les graphiques.

Remarque :

Le but de l'évaluation n'est pas de comparer les différents processus l'un par rapport à l'autre, mais plutôt d'évaluer un ou plusieurs processus afin de déceler les pratiques à améliorer et celles susceptibles de servir de tremplin à la démarche d'amélioration. Ainsi, le mode de cotation adopté permettra de dessiner des histogrammes représentant la contribution de

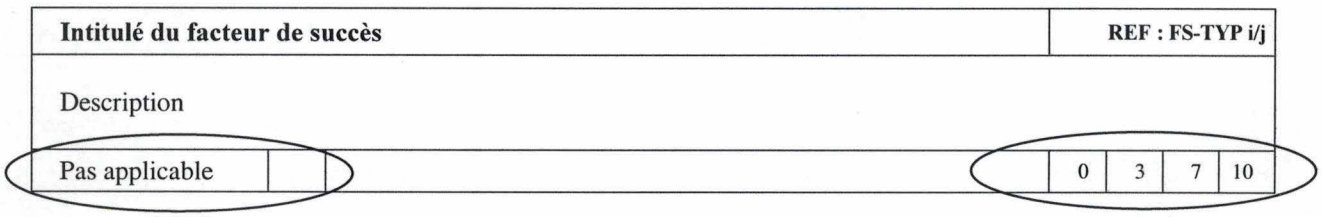
chaque pratique à la réalisation de l'objectif du processus d'une part, et le degré d'institutionnalisation de ces pratiques dans l'entreprise d'autre part.

La cote obtenue par chaque question est cumulée avec celle que cette même question a obtenu auprès des autres membres de l'échantillon. On peut ainsi facilement calculer la valeur moyenne d'une question et d'une pratique. La cote globale du processus est alors calculée en ajoutant les points qui ont été attribués aux délivrables.

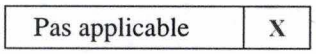
1.4.2 Cotation des facteurs de succès

Les facteurs de succès sont évalués sur base des réponses fournies par les membres de l'organisation à un formulaire écrit qui leur est remis lors des entretiens d'évaluation des processus.

Ils peuvent également être évalués par d'autres membres de l'organisation afin d'avoir un échantillon plus représentatif et donc une représentation fidèle de sa réalité quotidienne.

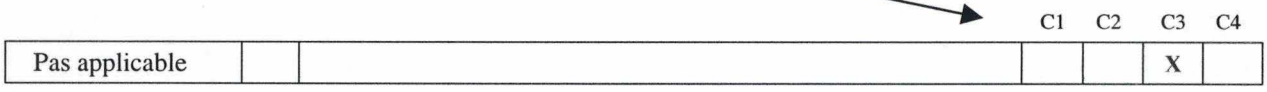


Les points qui ne peuvent être évalués soit parce que la personne ne dispose pas de l'information, soit parce que cela ne concerne pas l'entreprise, sont marqués *Pas applicable*.



Dans les autres cas, les facteurs de succès peuvent être évalués sur une échelle de quatre valeurs qui correspondent respectivement à :

(C1) **pas du tout**
(C2) **un peu**
(C3) **le plus souvent**
(C4) **tout à fait**



Ainsi, une affirmation pour laquelle la personne évaluée voudrait exprimer qu'elle est *généralement* d'actualité dans l'entreprise serait complétée comme ci-dessus.

Une valeur est associée à chaque niveau d'implémentation des facteurs de succès, ce qui permet de définir des tendances par type de facteurs de succès ou pour l'ensemble de ces

facteurs. Toute question jugée «**non applicable**» sera extraite du calcul grâce à une règle de trois.

Le but n'est pas d'analyser chaque facteur de succès dans le détail, mais plutôt d'obtenir une vue d'ensemble de la situation de l'entreprise et du support que celle-ci peut offrir pour la réalisation des objectifs de ses processus.

Ainsi, le mode de cotation adopté permettra de dessiner un histogramme comparant le niveau d'implémentation des quatre types de facteurs de succès (organisation, gestion, ressources humaines et technologie).

CHAPITRE 2. Analyse des exigences de l'utilisateur

2.1. Fonctionnalités de l'application

L'application à développer doit être un outil de représentation du modèle *OWPL* permettant la gestion du modèle et des éléments qui le composent. Ces éléments sont:

- le Glossaire du modèle *OWPL*
- les documents décrivant le modèle *OWPL* lui-même
- les questionnaires utilisés dans le cadre des évaluations basées sur le modèle *OWPL.*

Cette application doit aussi intervenir dans la procédure d'évaluation dans une entreprise, notamment grâce aux questionnaires cités plus-haut. L'aspect évolution et amélioration des pratiques logicielles dans les PME wallonnes étant plus important que l'évaluation elle-même, l'outil doit permettre de faire une correspondance entre le niveau des pratiques évaluées et les conséquences que ce niveau peut avoir sur la réalisation des objectifs, notamment par des propositions des recommandations qui doivent se trouver dans le rapport d'évaluation.

2.1.1 Le Glossaire

Le glossaire est la représentation de tous les termes présents dans le glossaire du modèle *OWPL* et offrant les fonctionnalités suivantes :

- Création d'un nouveau terme
- Consultation d'un terme
- Modification de la définition d'un terme
- Suppression d'un terme

Les différentes modifications au niveau du glossaire ont une répercussion directe sur le modèle et les questionnaires qui utilisent également certaines de ces définitions.

2.1.2 Le modèle *OWPL*

A n'importe quel niveau du modèle (processus, pratiques et facteurs de succès), il faut prévoir les opérations suivantes :

- création d'une nouvelle occurrence
- consultation d'une occurrence
- modification d'une occurrence
- suppression d'une occurrence

Il faut évidement tenir compte, lors des différentes opérations, de la cohésion interne du modèle et de son intégrité.
Les Sorties d'une pratique pouvant être les Entrées d'une autre, la suppression d'une pratique ou la modification de ses Entrées et Sorties peut avoir un effet sur les pratiques qui lui sont liées.

2.2. Opérations sur les éléments du modèle :

Dans le modèle *OWPL*, chaque élément est identifié de façon unique par une référence définie de manière arborescente sur la structure du modèle.

2.2.1 Processus

Il s'agit de gérer les processus du modèle *OWPL*:

- Création d'un nouveau processus
- Consultation d'un processus
- Suppression d'un processus
- Modification d'un processus

Concernant la modification, Il peut s'agir ici d'une modification de la description, de l'objectif ou, à un niveau inférieur, d'une modification concernant les pratiques du processus :

- Ajout d'une pratique
- Suppression d'une pratique

2.2.2 Pratiques

Il s'agit de la gestion des pratiques que comprennent les processus du modèle *OWPL* . Egalement, la numérotation des pratiques d'un processus doit à tout moment garder la logique arborescente et croissante, même après suppression ou ajout d'une pratique, tel que défini dans le modèle *OWPL*.
Les opérations suivantes sont donc nécessaires :

- Création d'une pratique
- Consultation d'une pratique
- Suppression d'une pratique
- Modification d'une pratique

Concernant la modification, Il peut s'agir ici d'une modification de l'objectif ou du poids d'une pratique ou, à un niveau inférieur, d'une modification concernant les composantes d'une pratique :

- Ajout d'un délivrable
- Suppression d'un délivrable
- Ajout d'une ressource
- Suppression d'une ressource

- Ajout d'un questionnaire
- Suppression d'un questionnaire

Concernant les délivrables, le système doit prendre en compte le fait qu'il y a un lien entre certains délivrables, qui sont sortie d'une pratique et entrée d'une autre. Ces liens ne sont pas mis en évidence dans le modèle.
Exemple : le cahier des charges, output de l'analyse des exigences, est un input pour l'analyse fonctionnelle.

2.2.3 Facteurs de succès

Il s'agit ici de gérer les facteurs de succès qui supportent les processus dans une entreprise. Il y a plusieurs catégories de facteurs de succès. A ce niveau, l'application à faire doit déjà prévoir la possibilité de :

- Création d'une catégorie
- Consultation d'une catégorie
- Modification d'une catégorie
- Suppression d'une catégorie

2.2.4 catégories de facteurs de succès

Il s'agit du traitement des facteurs de succès des catégories actuelles du modèle. Comme indiqué ci-dessus, ces catégories peuvent être crées, consultées, modifiées ou supprimer. Le traitement réservé aux facteurs de succès des catégories actuelles est valable pour des éventuels facteurs de succès des catégories qui seront créées plus tard.

Ainsi l'application doit prévoir, pour chaque catégorie de facteur de succès, les opérations suivantes :

- Création d'un facteur de succès
- Consultation d'un facteur de succès
- Modification d'un facteur de succès
- Suppression d'un facteur de succès

2.2.5 Les questionnaires

Les questionnaires sont utilisés dans le cadre des évaluations suivant le modèle *OWPL*, et sont modulables suivant des caractéristiques spécifiques à une entreprise. D'une part, il y a les questionnaires sur les processus, et d'autre part, il y a les questionnaires sur les facteurs de succès. Ces questionnaires sont définis suivant la structure du modèle *OWPL* et

utilisent certains termes du glossaire. Etant donné que les réponses à certaines questions dénotent une faiblesse ou une force, le système doit pouvoir proposer des recommandations en fonction des réponses reçues.

Ces recommandations sont faites sur base du niveau des pratiques constaté lors de l'évaluation (présence ou non de certains délivrables, nombre d'Entrées ou de Sorties d'une pratique,...).

Opérations à prévoir dans l'application :

Questionnaire des processus :

- Création d'une question
- Modification d'une question

Il peut s'agir à ici de la modification du libellé de la question ou, à un niveau inférieur, de la modification des propositions de réponses relatives à une question et de leurs cotes attribuées. Les opérations sont donc :

- ajout d'une proposition de réponse
- modification d'une proposition de réponse et/ou des cotes attribuées
- Consultation d'une question
- Suppression d'une question

Questionnaire des facteurs de succès :

- Création d'une question
- Modification d'une question

Il peut s'agir ici de la modification du libellé d'un facteur de succès ou de la modification de sa cotation.

- Consultation d'une question
- Suppression d'une question

2.3. Exigences non fonctionnelles

L'application étant destinée à être utilisée dans des entreprises dont on veut évaluer le processus logiciel, elle doit pouvoir être facilement portable et configurable. L'interface doit permettre une utilisation facile, car le bon déroulement d'une démarche qualité dans une entreprise dépendra du niveau d'implication de ses employés.

Il faudrait prévoir une option pour une éventuelle version bilingue de l'application. Il faudrait prévoir une certaine adaptabilité par rapport aux entreprises, notamment sur la nature de

certains délivrables. Certaines entreprises ont, par exemple, une gestion plus particulière de la documentation.
Il faudrait une gestion plus particulière des facteurs de succès. Ces derniers sont évalués de manière globale, sans correspondance explicite entre facteurs de succès et processus particuliers.

CHAPITRE 3. Modélisation des objets du système

Pour modéliser les objets du système à réaliser, nous avons choisi le modèle entité-association pour la clarté de la signification des objets utilisés. Pour avoir un aperçu de ce qu'est le modèle entité-association on peut se référer à ce petit résumé tiré de [5] : "*La réalité à représenter, un domaine d'application, est perçue sous la forme d'ensembles d'entités. Les entités sont en association les unes avec les autres. Les entités d'une classe sont du même type. Elles sont caractérisées par des attributs qui décrivent leurs propriétés intrinsèques. Les types d'associations sont caractérisés par leur classe, qui indique combien d'entités d'un type sont associés à une entité de l'autre type, et par leur caractère obligatoire ou facultatif pour les entités associées. Un type d'entités a un identifiant, généralement constitué d'attributs.* "

Dans cette partie, nous définissons, comme indiqué plus-haut, la notion à modéliser, puis, à l'aide d'un petit tableau, nous présentons les attributs de l'objet modélisé: La colonne NOM mentionne le nom de l'attribut. La colonne ID indique que l'attribut est un identifiant s'il contient le chiffre 1 (non significatif), et la colonne SEMANTIQUE indique le sens de chaque attribut.

Toutes les définitions et descriptions utilisées dans cette partie ont été tirées de [1] qui est la référence de la description du modèle OWPL. La figure 3 illustre le modèle Entité-Association du modèle OWPL.

3.1. Schéma Conceptuel de la base des données

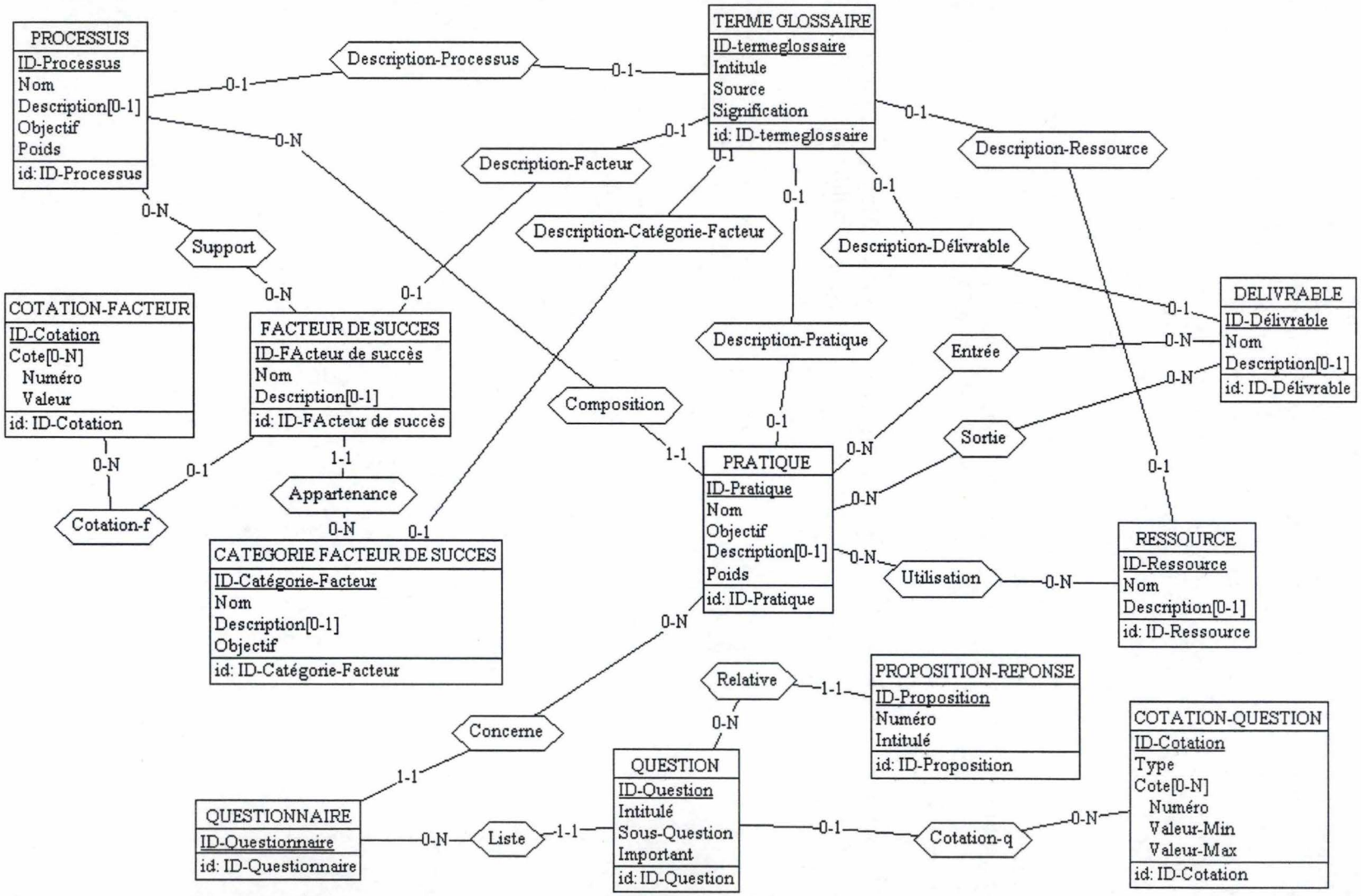


Figure 3 Schéma conceptuel du modèle OWPL

3.2. Les types d'entités

3.2.1 Processus

Un processus est un *ensemble structuré de pratiques nécessaires à la réalisation d'un objectif commun clairement défini.*
Le processus est défini en fonction de l'objectif global de l'entreprise. S'appuyant sur des *facteurs de succès*, il permet d'organiser différentes pratiques en un tout cohérent et contribue ainsi à la réalisation de l'objectif de l'entreprise.
Chaque processus est défini par son objectif et la liste des pratiques qui le composent. (Cfr. Figure 2: Structure du modèle OWPL[1]).

NOM	ID	SEMANTIQUE
ID-Processus	1	Identifiant du processus dans le modèle OWPL
Nom		Nom du processus
Description[0-1]		Description du processus
Objectif		Objectif du processus
Poids		Poids du processus

3.2.2 Pratique

Une pratique est une *activité d'ingénierie qui contribue à la réalisation de l'objectif d'un processus par la création d'un délivrable ou l'amélioration de la capacité du processus.*
Chaque pratique est caractérisée par son objectif, ses entrées, ses sorties, ses ressources et son poids.
Les pratiques sont organisées de façon à faciliter leur articulation autour d'un objectif central, celui du processus dont elles font partie. Une pratique est composée de 5 éléments : son objectif, ses entrées, ses sorties, les ressources qui lui sont nécessaires et son poids dans la réalisation de l'objectif du processus auquel elle appartient (Cfr. Figure 2: Structure du modèle OWPL[1]).

NOM	ID	SEMANTIQUE
ID-Pratique	1	Numéro identifiant de la pratique dans le modèle OWPL
Nom		Nom de la pratique
Description[0-1]		Description de la pratique
Objectif		Objectif de la pratique
Poids		Poids de la pratique

3.2.3 Catégorie facteur de succès

Les facteurs de succès sont des *éléments d'environnement qui favorisent la mise en place d'un support permettant une exécution optimale des processus.*
Ils sont regroupés en quatre catégories : l'organisation dans laquelle se déroulent les processus, la politique de gestion mise en place, les ressources humaines mobilisées et les moyens techniques utilisés. (Cfr. Figure 2 : Structure du modèle OWPL).

NOM	ID	SEMANTIQUE
ID-Catégorie facteur	1	Numéro identifiant de la catégorie de facteur de succès dans le modèle OWPL
NOM		Nom de la catégorie
Objectif		Objectif de la catégorie
Description[0-1]		Description de la catégorie

3.2.4 Facteur de succès

Ce type d'entité regroupe ici tous les facteurs de succès. Chaque facteur de succès a un numéro identifiant dans le modèle OWPL, un nom et est éventuellement décrit dans le glossaire.

NOM	ID	SEMANTIQUE
ID-Facteur de succès	1	Numéro identifiant du facteur de succès dans le modèle OWPL
NOM		Intitulé du facteur de succès
Description[0-1]		Description du facteur de succès

3.2.5 Cotation-facteur

Les facteurs de succès sont évalués dans le cadre du modèle OWPL à l'aide d'un questionnaire comme nous l'avons vu plus haut. Le type d'entité COTATION-FACTEUR regroupe l'ensemble des cotes qui sont attribuées aux différentes cases du questionnaire relatif au facteur de succès concerné.

NOM	ID	SEMANTIQUE
ID-Cotation	1	Numéro identifiant
Cote[0-N] Numéro Valeur		Cotes du facteur de succès Numéro d'ordre de la cotation Valeur de la cotation

3.2.6 Ressource

Au point II.3.1.2, nous parlons des ressources qui sont utilisés dans une pratique. Ce type d'entité représente donc ces ressources, dont chaque occurrence a un nom et une description. Etant donné que des ressources différents peuvent avoir un même nom, un numéro identifiant est ajouté à chaque ressource.

NOM	ID	SEMANTIQUE
ID-Ressource	1	Numéro identifiant de la ressource
NOM		Nom de la ressource
Description[0-1]		Description de la ressource

3.2.7 Délivrable

Au même titre que les ressources, une pratique emploie des délivrables en entrée pour son bon déroulement, et fournit des délivrables en sortie. Ces délivrables sont caractérisés par un numéro identifiant pour les mêmes raisons que les ressources, ont un nom et une description.

NOM	ID	SEMANTIQUE
ID-Délivrable	1	Numéro identifiant du délivrable
NOM		Nom du délivrable
Description[0-1]		Description du délivrable

3.2.8 Questionnaire

Ce type d'entité représente un ensemble de questions relatives à une pratique. Dans les documents de présentation du modèle OWPL, " *le questionnaire* " est l'ensemble des questions que l'on pose lors d'une interview. Cette dernière notion est différente de ce type d'entité. Néanmoins, l'ensemble des questionnaires de toutes les pratiques forme *le questionnaire* du modèle OWPL.

NOM	ID	SEMANTIQUE
ID-Questionnaire	1	Numéro identifiant du questionnaire dans le modèle OWPL

3.2.9 Question

Comme expliqué dans le point précédents, ce type d'entité regroupe toutes les questions qui peuvent être posées dans le cadre de l'évaluation d'une pratique. Une question a donc un numéro identifiant, un intitulé (le libellé de la question), une éventuelle sous-question et n sait voir si cette question est importante ou non.

NOM	ID	SEMANTIQUE
ID-Question	1	Numéro identifiant du facteur de succès dans le modèle OWPL
Intitulé		Intitulé de la question dans le modèle OWPL
Sous-Question[0-1]		Question additionnelle qu'on pose lors de l'interview
Important		Vrai si la sous-question est importante

3.2.10 Proposition-réponse

Dans cette entité sont regroupés toutes les propositions de réponse qu'on propose dans un questionnaire des pratiques. A chaque proposition de réponse est associée une question.

NOM	ID	SEMANTIQUE
ID-Proposition	1	Numéro identifiant d'une proposition de réponse
Numéro		Numéro d'ordre de la proposition de réponse
Intitulé		Intitulé de la proposition de réponse

3.2.11 Cotation-question

Dans cette entité, on retrouve la façon de coter une pratique. A chaque type de cotation est associée une cotation. Une pratique peut être cotée de deux façons (voir supra II.1.4.1), cela est indiqué par l'attribut type. Une sous-question peut être posée lors de l'interview. Il y a également une liste des cotes relatives à chaque proposition de réponse.

NOM	ID	SEMANTIQUE
ID-Cotation	1	Numéro identifiant de la Cotation
Type		Le type de la Cotation (MAXIMUM ou TOTAL)
Cote [0-N] Numéro Valeur-Min Valeur-Max		 Numéro d'ordre relatif à la ligne de Cotation Valeur minimale de la cote Valeur maximale de la cote

3.2.12 Terme glossaire

Un glossaire accompagne le modèle OWPL. Dans ce glossaire sont expliqués quelques termes techniques utilisés dans le modèle. Ce type d'entité reprend tous les termes du glossaire. Ces termes sont caractérisés par un numéro identifiant, un intitulé, une source et une signification.

NOM	ID	SEMANTIQUE
ID-termeglossaire	1	Identifiant du terme du Glossaire
Intitulé	1'	Intitulé du terme dans le glossaire du modèle OWPL
Source		Source de la signification
Signification		Signification du terme dans le glossaire du modèle OWPL

3.3. Les type d'associations

Tous les types d'associations sont présentés de la manière suivante : on présente d'abord les entités concernés avec leurs cardinalités[5]. Nous donnons ensuite une sémantique pour chaque type d'association.

[5] Les cardinalités pour les types d'associations constituent leur classe, elles indiquent combien d'entités d'un type sont associés à une entité de l'autre type, et indiquent le caractère obligatoire ou facultatif pour les entités associées.

3.3.1 Support

Entités concernées : PROCESSUS [0-N]
FACTEUR DE SUCCES[0-N]
Sémantique : Dans le modèle OWPL, tous les facteurs de succès supportent, de manière générale, tous les processus. Dans quelques cas, on peut établir un lien direct entre un facteur de succès et un processus. C'est ce lien qui est matérialisé par ce type association.

3.3.2 Appartenance

Entités concernées : CATEGORIE FACTEUR DE SUCCES [0-N]
FACTEUR DE SUCCES[1-1]
Sémantique : Un facteur de succès appartient à une catégorie déterminée de facteur de succès dans le modèle OWPL.

3.3.3 Composition

Entités concernées : PROCESSUS [0-N]
PRATIQUE[1-1]
Sémantique : Un processus est composé de plusieurs pratiques dans le modèle OWPL.

3.3.4 Entrée

Entités concernées : PRATIQUE [0-N]
DELIVRABLE[0-N]
Sémantique : Un délivrable peut être une entrée pour une pratique dans le modèle OWPL.

3.3.5 Sortie

Entités concernées : PRATIQUE [0-N]
DELIVRABLE[0-N]
Sémantique : Un délivrable peut être une sortie pour une pratique dans le modèle OWPL.

3.3.6 Utilisation

Entités concernées : PRATIQUE [0-N]
RESSOURCE[0-N]
Sémantique : Une ressource peut être utilisée par une pratique dans le modèle OWPL.

3.3.7 Concerne

Entités concernées : PRATIQUE [0-N]
QUESTIONNAIRE [1-1]

Sémantique : Un questionnaire[6] concerne une pratique dans le modèle OWPL.

3.3.8 Liste

Entités concernées : QUESTIONNAIRE [0-N]
QUESTIONS[1-1]
Sémantique : Un questionnaire[7] est composé d'un ensemble des questions dans le modèle OWPL.

3.3.9 Relative

Entités concernées : QUESTION [0-N]
PROPOSITION REPONSE[1-1]
Sémantique : Une proposition de réponse est relative à une seule question.

3.3.10 Cotation-q

Entités concernées : COTATION-QUESTION[0-N]
QUESTION[0-1]
Sémantique : Une question est liée à un mode de cotation bien déterminé .

3.3.11 Cotation-f

Entités concernées : COTATION-FACTEUR[0-N]
FACTEUR DE SUCCES[0-1]
Sémantique : Un facteur de succès est lié à un mode de cotation bien déterminé .

Toutes les associations qui suivent (de la forme Description-Entité) ont comme caractéristique commune :

Entités concernées : TERME-GLOSSAIRE [0-1]
ENTITE[0-1]

[6] L'entité QUESTIONNAIRE telle que définie au point 3.1.8

Sémantique : Un terme du glossaire peut donner la description d'un terme utilisé dans le modèle OWPL.
Il s'agit des associations :

- Description-Processus
- Description-Facteur
- Description-Catégorie Facteur
- Description-Pratique
- Description-Ressource
- Description-Délivrable

3.4. Contraintes supplémentaires du modèle Entité-Association

- Soit une occurrence X de Question, N occurrences de Proposition-réponse et une occurrence Z de Cotation-pratique. S'il existe une occurrence de l'association « Cotation-q » entre X et Z (X est une question qui a une cotation particulière), et il existe N occurrences de l'association « Relative » entre X et les N Propositions-réponses(X est une question qui a N propositions de réponse), le nombre de cotes de Z est égal à N (la cotation Z de la question X est composée de N cotes qui correspondent chacune à une proposition de réponse).

- Les identifiants des Pratiques d'un même processus sont structurés de manière croissante et conformément au modèle OWPL.

  Exemple : Le processus "Capitalisation des acquis" est composé des trois pratiques suivantes :
  "Analyse des projets antérieurs" avec comme identifiant "CPTL/PR01/03"
  "Définition/adaptation du cadre d'apprentissage " avec comme identifiant " CPTL/PR02/03"
  "Information continue sur le manuel qualité" avec comme identifiant "CPTL/PR03/03"

  Si pour une raison ou une autre on supprime la deuxième pratique, on aura les deux pratiques suivantes :
  "Analyse des projets antérieurs" avec comme identifiant "CPTL/PR01/02"
  "Information continue sur le manuel qualité" avec comme identifiant "CPTL/PR02/02".

- Pour chaque attribut Description[0-1] présent dans une entité ayant une association de type « Description-Entité » avec l'entité Terme-Glossaire, il existe une contrainte d'exclusion entre cet attribut et l'association « Description-Entité ». En d'autres termes, il ne peut exister une description d'un terme du modèle OWPL si cette description est donnée dans le glossaire.

III. CONCEPTION DE L'APPLICATION

CHAPITRE 4. Choix technologiques

Une des exigences les plus importantes de l'application à développer est la **portabilité**. C'est dans ce cadre que notre attention s'est d'abord focalisée sur le choix d'une architecture pour notre base des données.

Devant la multitude des solutions possibles, une procédure par élimination des technologies les plus compliquées s'est vite imposée, le critère de portabilité étant toujours de mise.

Etant donné que l'application à développer est censée être utilisée lors des évaluations dans des entreprises diverses, chacune ayant sa propre plate-forme de fonctionnement, nous avons éliminé de prime abord tous les systèmes de gestion des bases de données tels que Oracle et Access. Ces derniers ont l'avantage d'être très efficaces et offrent beaucoup plus de sécurité. En les adoptant on résoudrait une grande partie des problèmes liés à la gestion de la base des données. Par contre, ce qui constitue leur faiblesse, c'est surtout la lourdeur de leur procédure d'installation et le coût que leur installation peut occasionner dans une PME Wallonne juste pour procéder à une évaluation des pratiques logicielles. Cette dernière raison va d'ailleurs dans le sens d'une des raisons d'être du modèle OWPL, à savoir le coût élevé des évaluations suivant les modèles SPICE et CMM pour les PME Wallonnes.

Il fallait donc opter pour une solution d'une base des données portable et facilement utilisable. C'est pourquoi nous avons opté pour des fichiers XML. Les premiers points de ce chapitre présentent la technologie XML: le langage, les parseurs, le DOM, le SAX, et les feuilles de style.

Nous réfléchirons ensuite sur la technologie à utiliser pour manipuler cette base des données en partant de Javascript jusqu'aux applets java signées, pour terminer avec la solution retenue pour notre application, et l'évaluation de la pertinence des choix effectués.

4.1. Des Documents XML comme base des données

4.1.1 XML

4.1.1.1 Historique

Le XML (*eXtensible Markup Language*) est un langage de marquage constitué des balises tout comme HTML[7]. Il se situe à mi-chemin entre le SGML et le HTML que l'on rencontre tous les jours sur l'Internet.

Tout comme son cousin le HTML, le XML est directement issu du SGML. Cependant, il s'en rapproche davantage dans le sens où l'on peut dire que XML est une forme simplifiée de SGML, car ce dernier est trop complexe pour s'afficher sur le WEB. Une présentation du SGML est faite au point suivant.

[7] HyperText Markup Language, langage d'édition des pages web.

Le XML n'est pas un langage sémantiquement figé comme peut l'être le HTML. En effet, le HTML utilise un nombre défini des balises pour décrire les pages web, mises à part quelques balises propres à certains navigateurs (Exemple : la balise <marquee> </marquee> pour Internet Explorer 5.0 qui fait défiler du texte sur l'écran). Au contraire, le XML est un langage ouvert. C'est-à-dire que l'auteur d'un document XML peut créer ses propres balises. Par exemple, la balise <INSTRUMENT> peut être définie pour désigner un instrument de musique. Cela s'écrirait de la façon suivante :

```
<INSTRUMENT>Guitare</INSTRUMENT>
```

Dans un document XML, comme pour le SGML, on s'efforcera de ne pas tenir compte de la mise en forme mais seulement du contenu de celui-ci, la mise en forme étant réalisée par une feuille de style. Pour le XML, on peut utiliser des feuilles de style XSL (eXtensible Style Language) ou CSS (Cascading Style Sheets). Les feuilles CSS sont aussi utilisées avec du HTML. Actuellement, il existe très peu de navigateurs XML[8] de qualité. Internet Explorer , à partir de sa version 4.0 et Netscape Navigator dans sa version libre Mozilla permettent de reconnaître des documents XML. Si Internet Explorer a amélioré la prise en charge de XML dans sa version 5.0, Mozilla n'offre pas encore de vraie stabilité. Le navigateur XML le plus intéressant est le navigateur InDelv XML qui offre la prise en charge la plus complète du XSL.

Dès lors, ont peut imaginer d'innombrables possibilités que nous offre le langage XML; par exemple dans l'échange de données informatisées par des liaisons EDI (Electronic Data Interchange).

Prenons par exemple le secteur automobile, on pourrait employer les balises suivantes pour décrire un véhicule:

```
<TYPE MINE>, <CHASSIS>, <MOTEUR>, <COULEUR>, …
```

Les données pouvant être structurées, comme on définirait un enregistrement Moteur dans tout langage de programmation ou bien dans une base de données, en XML on écrira :

```
<MOTEUR>
        <PUISSANCE>…..</PUISSANCE>
        <CYLINDREE>…..</CYLINDREE>
        <NB SOUPAPES>…..</NB SOUPAPES>
</MOTEUR>
```

Puisque, dans un document XML, l'information pertinente est marquée par des balises portant des noms significatifs, il sera plus facile de retrouver l'information en s'appuyant sur les noms de ces balises pour effectuer des recherches. En effet, en HTML on ne sait faire actuellement que de la recherche plein-texte (*full-text*), ce qui la plupart du temps nous retrouve beaucoup de documents qui n'ont rien à voir avec notre recherche de départ, c'est ce que l'on appelle le *bruit*.

Dans notre exemple, Un moteur de recherche pourrait aisément retrouver tous les documents traitant des moteurs 16 soupapes équipant un châssis d'un type donné. On serait

[8] Un navigateur XML est comme un navigateur HTML et est destiné à afficher des documents XML.

alors sûrs que les réponses correspondraient exactement à nos critères de sélection. Ainsi, les autres documents ne traitant qu'un seul de ces deux critères ne seraient pas retournés. De même, l'utilisateur obtenant ces réponses pourrait demander à son application de les classer par puissance de ces moteurs.

4.1.1.2 Syntaxe XML

Soit le fichier XML suivant représentant un carnet d'adresses:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-Simple carnet d'adresse -->
<carnet-adresses>
   <entrée>
         <nom>Cathy YELEPENE</nom>
         <adresse>
               <rue>11 rue de la paix</rue>
               <code-postal> 4100 </code-postal>
               <localité>Seraing</localité>
               <pays>Belgique</pays>
         </adresse>
         <tel courant="vrai">043382907</tel>
         <tel>0495151206</tel>
         <email href="mailto:cys@hotmail.com"/ >
   </entrée>
   <entrée>
          <prénom>Jimmy</prénom><nom>BATILA</nom>
          <tel>0012173551611</tel>
          <email href="mailto:jimbat@hotmail.com"/ >
   </entrée>
</carnet-adresses>
```

Ce document XML est avant tout un simple document texte. Il est constitué de données caractères et des balises. Les données caractères contiennent l'information et les balises renseignent sur la structure du document. Ce même carnet d'adresse peut être représenté par le fichier texte ci-dessous.

```
Cathy YELEPENE
11 rue de la paix
4100 Seraing
Belgique
043382907
0495151206
cys@hotmail.com
Jimmy BATILA
0012173551611
jimbat@hotmail.com
```

Si pour l'humain le deuxième texte est plus clair et facilement lisible, le premier avec des balises est plus lisible pour l'ordinateur du point de vue de la sémantique accordée à chaque information. C'est pourquoi les balises sont importantes : le texte est découpé dans ses parties constructives, permettant au programme de les traiter comme il convient.

L'élément constitutif d'un document XML repose sur un nom et un contenu :

```
<tel>0495151206</tel>
```

Le contenu d'un élément(`0495151206`) est compris entre la balise de début et la balise de fin. La balise de début(`<tel>`) est le nom de l'élément entre crochets, et la balise de fin (`</tel>`) est symbolisée par l'ajout d'une barre oblique à la balise d'ouverture.

Les noms du XML doivent commencer par un caractère alphabétique ou un caractère de soulignement, les autres caractères pouvant être alphanumériques, le soulignement, le point ou le trait d'union. Les espaces ne sont pas autorisés, et les noms ne peuvent commencer par la séquence "xml", réservée à XML.

Il est possible d'ajouter des informations supplémentaires aux éléments sous forme d'attributs. Ces derniers ont un nom et une valeur. Les noms d'attributs suivent les mêmes règles que les noms d'éléments. Le nom d'attribut est séparé par le signe =, et la valeur est indiquée entre guillemets droits, les guillemets pouvant être remplacés par un simple apostrophe.

Un exemple d'utilisation d'un attribut : `<tel courant="vrai">043382907</tel>`
Le nom de l'attribut ici est "courant" , et sa valeur est "vrai".

Les éléments qui n'ont pas de contenu sont dits éléments vides, et sont utilisés uniquement pour la valeur de leurs attributs. L'élément `<email href="mailto:jimbat@hotmail.com"/>` en est un exemple. Un élément peut en contenir d'autres, il s'agit des éléments imbriqués qui peuvent à leur tour contenir du texte ou d'autres éléments.

Un document XML peut être hiérarchisé dans une arborescence. Il n'y a aucune limite à la profondeur de l'arbre, et les éléments peuvent se répéter. Le fichier XML ci-haut possède deux `entrée` dans l'élément `carnet-d'adresse`. L'entrée pour `Cathy YELEPENE` comporte deux éléments pour le téléphone. On peut représenter ce carnet d'adresse par l'arbre ci-dessous.

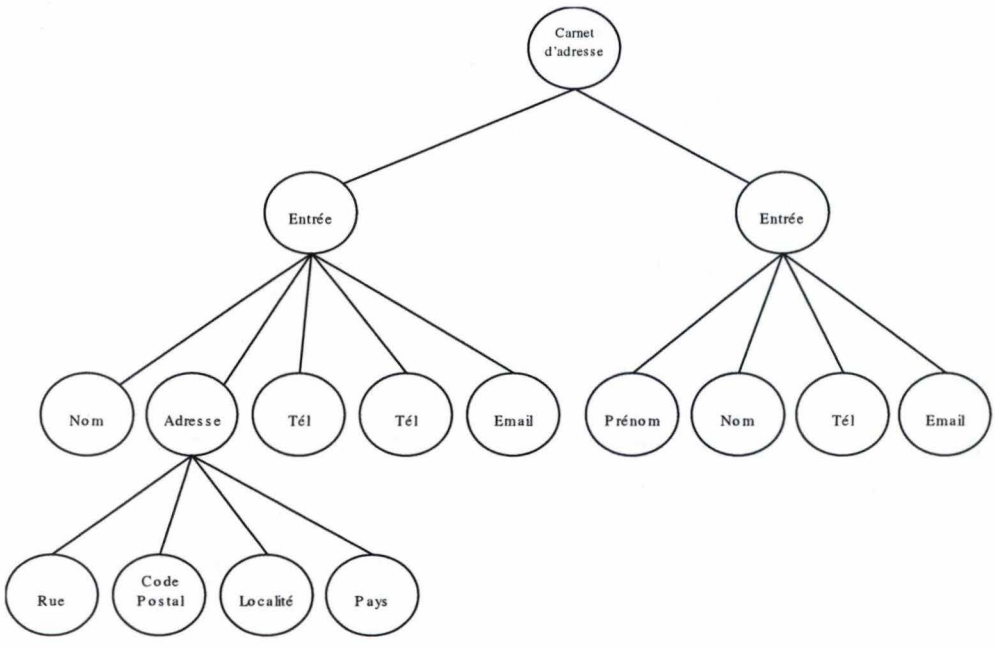


Figure 4 L'arbre du carnet d'adresse

Quand un élément est imbriqué dans un autre, c'est un élément enfant contenu dans un élément parent. La balise de fermeture d'un enfant ne peut pas se situer après la balise du parent. Dans l'exemple qui suit, l'élément `nom` a deux enfants, les éléments `prénom` et `nomf` :

```
<nom>
        <prénom>Jimmy</prénom>
        <nomf>BATILA</nomf>
</nom>
```

A la racine d'un document XML, il ne peut y avoir qu'un seul et unique élément. Cela signifie que tous les éléments d'un document XML sont les enfants d'un seul élément. Les commentaires en XML sont placés entre `<!--` et `-->`. La ligne `<!–Simple carnet d'adresse -->` dans notre exemple est donc un commentaire.

Enfin, tout document XML doit commencer par la déclaration XML qui définit un document comme étant un document XML. La version du XML utilisé est indiquée, avec ici la version 1.0. Dans notre exemple, il s'agit de `<?xml version="1.0" encoding="ISO-8859-1"?>`. Ces différentes versions peuvent être rejetées par un parseur particulier. La déclaration XML peut aussi comporter d'autres attributs comme `encoding`, indiquant ici le jeu de caractères utilisé.

Cette description de la syntaxe XML est loin d'être complète, mais assez pour comprendre la syntaxe qui sera utilisée dans notre application. D'autres notions, plus complexes sont expliquées dans la recommandation XML 1.0 du consortium W3C[9]. Le lecteur intéressé peut en trouver un exemplaire sur **http://www.w3.org/TR/1998/REC-xml-19980210**.

4.1.2 SGML

Le SGML (*Standard Generalized Markup Language*) défini dans la norme ISO 8879 est une norme internationale adoptée dès 1986 pour apporter une réponse à des problèmes de gestion de documents : utilisateurs hétérogènes, données pouvant être présentées sous plusieurs formats (écran, papier, écriture braille,...)

Cette norme ISO est ainsi fondée sur la distinction fondamentale entre contenu et présentation physique. Le SGML repose sur le principe de structure logique d'une texte.

Exemple : Imaginons un ouvrage qui porte le titre "Mon premier Livre".
On décrira ce titre avec le symbolisme suivant :

```
<titre>Mon premier Livre</titre>
```

Tous les attributs typographiques ne sont pas ici représentés, seule une balise nous renseigne sur la nature de l'objet représenté par ce texte. Le texte est systématiquement marqué par une chaîne de balises entre crochets qui énoncent seulement que l'élément dans la balise `<titre>` est le titre de l'ouvrage. Si on généralise cet exemple, on peut ainsi représenter n'importe quel attribut d'un livre comme `<sous-titre></sous-titre>`,`<auteur></auteur>`,etc. La représentation typographique est laissée à l'outil de mise en page qui lui, donne une forme physique.

[9] W3C, World Wide Web Consortium (http://www.w3.org/TR/REC-DM-Level-1). Le W3C est une organisation chargée du développement et de la maintenance de la plupart des normes du web comme le HTML.

Pour ce faire, on décrit une feuille de style pour la traduction physique de chaque balise. Pour le SGML, il s'agit de DSSSL (*Document Style Semantics and Specification Language*).

Avec le SGML, une fois q'un industriel a déterminé la structure de ses documents, il peut garder pendant longtemps la même structure tout en changeant au besoin, le traitement associé à chaque balise. La définition logique de son modèle de document permet de vérifier qu'un document respecte les spécifications.

Bref, concernant le XML et le SGML, le XML est un sous-ensemble du SGML. Tout ce qu'on peut faire sur des document avec le SGML, le XML permet de le faire avec en plus la facilité de les représenter, ce qui n'est pas toujours aisée avec le SGML. La supériorité du XML par rapport au HTML se situe au niveau de la création des balises plus significatives pour l'utilisateur et la présence d'une sémantique à l'intérieur d'un document.

4.1.3 DTD

DTD signifie *Document Type Definition*, il s'agit de la définition de type de document, c'est-à-dire la spécification de la structure d'un document XML. "*Elle fournit pour un document XML, la liste des éléments, des attributs, des notations et des entités que contient le document ainsi que les règles des relations qui les régissent*"[7].

Pour associer une DTD à un document XML, on peut soit l'insérer dans le document, soit faire une référence d'adresse URL[10] externe dans le document. Dans ce dernier cas, une DTD peut être partagée par plusieurs documents, et c'est la solution idéale pour spécifier la structure des documents XML destinés à être partagés par différents systèmes autonomes: un éditeur peut par exemple imposer une certaine structure pour tous les documents XML décrivant un livre qui arrivent dans son système.
On peut aussi se servir d'une DTD pour écrire les feuilles de style associées à des documents XML, sans prendre nécessairement connaissance du contenu de ces documents. Les feuilles de style sont traitées au point 4.1.6. Dans une DTD, les majuscules sont distinguées des minuscules.

Contrairement à un document bien formé (document respectant la syntaxe XML), un document valide (document conforme à sa DTD) ne permet pas d'utilisation des balises de manière arbitraire. Toutes la balises qu'on utilise doivent être présentes dans la DTD.

Prenons pour exemple, la DTD du glossaire simplifié du modèle OWPL: nous rappelons que notre glossaire est composé des termes, et un terme est défini par un intitulé, une signification et une source.

La DTD du document XML décrivant les termes du glossaire est le suivant:

```
<!DOCTYPE termeglossaires [
<!ELEMENT termeglossaires(terme*)>
<!ELEMENT terme(intitule, signification, source)>
<!ELEMENT intitule(#PCDATA)>
```

[10] URL(Uniform Resource Locator), nom universel associé à une page sur Internet. Un URL est composé de trois parties:le protocole utilisé, le nom DNS de la machine sur laquelle se trouve la page et un nom local indiquant spécifiquement la page.

```
<!ELEMENT signification(#PCDATA)>
<!ELEMENT source(#PCDATA)>
]>
```

La première ligne (`<!DOCTYPE termeglossaires [`) signifie que l'élément de premier niveau dans notre document doit être `termeglossaires`. Tout ce qui est entre les crochet constitue la définition de la structure.

`<!ELEMENT termeglossaires(terme*)>` signifie que l'élément `termeglossaires` a un élément enfant, `terme`. Le signe `*` après terme est un indicateur d'occurrence. Les indicateurs d'occurrences suivent le nom d'un élément. L'indicateur `*` signifie ici qu'on peut avoir zéro ou plusieurs termes dans le document. D'autres indicateurs d'occurrences sont `?` (un élément peut apparaître au plus une fois ou) et `+` (l'élément doit apparaître au moins une fois). Un élément sans indicateur d'occurrence signifie qu'il ne peut apparaître qu'une fois.
`<!ELEMENT terme(intitule, signification, source)>` signifie que l'élément terme contient trois éléments enfants : `intitule`, `signification` et `source`.

`<!ELEMENT intitule(#PCDATA)>`,`<!ELEMENT signification(#PCDATA)>` et `<!ELEMENT source(#PCDATA)>` signifient que les éléments `intitule`, `signification` et `source`, ne contiennent que du texte, pas d'autres éléments. `PCDATA` pour *Parser Character Data* (donnée textuelle analysable) signifie justement que entre les deux balises de l'élément concerné, on doit obligatoirement trouver du texte, et cet élément ne peut pas contenir des sous-éléments.

Un exemple de document XML valide par rapport à cette DTD:

```
<?xml version="1.0" standalone="yes"?>
<termeglossaires>
      <terme>
            <intitule>Codage</intitule>
            <signification>Traduction de la description technique du
système en code opérationnel, compte tenu de l'environnement
d'exploitation.[OWPL]</signification>
            <source>OWPL</source>
      </terme>
      <terme>
            <intitule>Acquis</intitule>
            <signification>Expérience accumulée dans le passe et
susceptible d'enrichir les projets en cours ou
futurs.[OWPL]</signification>
            <source>OWPL</source>
      </terme>
</termeglossaires>
```

4.1.4 Parseur XML

Quand on possède un fichier XML, si on veut en consulter le contenu, il faut passer par un programme dont la majorité du code servira à interpréter la syntaxe XML. Interpréter cette syntaxe n'est pas difficile, mais prend beaucoup de temps. C'est pourquoi les développeurs utilisent un parseur qui permet de valider la syntaxe du XML. Le terme parseur est emprunté à l'univers des compilateurs. En effet, un parseur est le module du compilateur qui permet de lire et d'interpréter le fichier source d'un langage de programmation. Le parseur

crée un arbre d'analyse qui est la représentation en mémoire du code source. Le compilateur utilise alors cet arbre d'analyse pour créer du code objet.

Les documents XML peuvent être bien formées ou valides. On dit qu'un document XML est bien formé quand il respecte la syntaxe XML, et on dit qu'il est valide quand il respecte la syntaxe et la structure définies dans une DTD.

Il existe des parseurs validant et des parseurs non validant. Les parseurs non validant vérifient juste la syntaxe XML et les parseurs validant savent valider des documents XML par rapport à leur DTD. L'avantage d'utiliser un parseur validant est qu'il permet de repérer des erreurs de structure sans qu'on écrive une ligne de code à cette fin.

Un application utilisant un parseur s'en sert pour accéder aux fichiers XML. Pour ce faire, une interface doit être définie entre le parseur et l'application. En général le parseur se charge de lire un fichier XML et de créer une copie de son arbre d'analyse en mémoire. L'application va manipuler cet arbre comme s'il s'agissait du document XML. Il existe deux façons d'interfacer un parseur et une application : les interfaces objets et les interfaces événementielles.

Quand on utilise une interface objet, le parseur crée explicitement un arbre d'objets contenant tous les éléments du document XML. Il s'agit ici de l'interface la plus naturelle, car l'arbre en mémoire correspond exactement au document XML. L'interface événementielle consiste à interfacer le parseur et l'application par l'intermédiaire des événements. Avec une telle interface, le parseur ne construit pas explicitement un arbre d'objets, il lit le document et génère des événements quand il rencontre des éléments comme des attributs ou du texte dans le fichier XML. Les deux types d'interface ne servent pas pour les mêmes objectifs. Les interfaces objets sont adaptées pour les applications qui manipulent des documents XML comme des navigateurs et des éditeurs. les interfaces événementielles sont adaptées pour les applications qui manipulent des documents dans une structure autre que XML, même si ces documents sont sauvegardés dans des fichiers XML.

4.1.5 DOM et SAX

Comme pour SQL et les bases de données relationnelles, il faut standardiser les interfaces parseur-application. L'avantage ici est qu'une application écrite avec un parseur peut être utilisée avec un autre parseur.

Même si dans la pratique, la standardisation n'est pas respectée à 100%, les deux sortes d'interfaces ont fait l'objet de standardisation.

Les interfaces objet sont connues sous le nom de DOM (*Document Object Model*) publié par le W3C. Les interfaces événementielles sont elles connues sous le nom de SAX (*Simple API for XML*), développées par les membres du groupe XML-DEV[11]. De nombreux parseurs comme ceux de IBM XML for Java ou SUN ProjectX prennent en charge les deux interfaces.

[11] Elles sont publiées par David Meggison (http://www.meggison.com/SAX).

4.1.5.1 DOM

Le DOM fournit des API abstraites pour la construction, l'accès et la manipulation de documents XML et HTML. Une implémentation du DOM dans un langage de programmation donné offre une API concrète.

L'objet central du DOM est le nœud (Node). C'est un objet générique dans l'arbre et la plupart des objets du DOM en sont dérivés. Il existe des versions spécialisées pour les éléments, les attributs, les entités , le texte, etc...

Les nœuds définissent différentes propriétés pour faciliter le parcours de l'arbre :

Par exemple, `Nodetype` est un code représentant le type d'objet défini dans le tableau non exhaustif suivant :

Types des nœuds du DOM

TYPE	CODE
Elément	1
Attribut	2
Texte	3
Commentaire	8
...	

`ParentName` est le parent, s 'il existe, de l'objet courant.
`ChildNode` est la liste des enfants de l'objet courant.
`firstChild` et `lastChild`, respectivement le premier enfant du nœud et le dernier enfant du nœud.

`Documentelement` est l'élément de premier niveau dans le document, et `doctype` est le type de document, qui sera précisé dans la version de niveau du DOM. `Document` est la racine, c'est l'objet précédant l'élément de premier niveau.

Après le parsing d'un document XML, le parseur DOM retourne un objet `Document`, il est alors possible d'accéder à la totalité de l'arborescence. Le parcours de l'arborescence s'effectue en utilisant un algorithme de récursivité qui visite tous les nœuds de l'arbre. DOM est une interface multi-plateformes et multi-langages. Il existe des version DOM pour Javascript, Java et C++, ceci grâce au fait que le W3C utilise OMG IDL[12] pour la spécification DOM.

4.1.5.2 SAX

Pour des applications qui veulent garder des documents dans une structure autre que XML, l'interface événementielle est plus adaptée. Celle-ci ne construit pas un arbre, mais déclenche des événements lors de la lecture d'un fichier XML. Ces événements ne sont pas déclenchés par l'utilisateur, mais par les structures rencontrées lors de la lecture du fichier. Il peut s'agir de la balise d'ouverture d'un élément, de la balise de fermeture, du contenu des éléments, des erreurs d'interprétations, etc.

[12] OMG IDL pour Object Management Group Interface Definition Language. C'est un langage de spécification des interfaces objet. Il décrit les méthodes et les propriétés d'un objet, sans se préoccuper de l'implémentation.

A la fin de la lecture du fichier, les informations reçues par l'application sont utilisées pour créer la structure des données propre à l'application. Même s'il y a assez d'information pour construire un arbre, cet arbre n'aura aucune utilité, car l'application est censée dans ce cas utiliser sa propre structure des données. C'est donc pourquoi un arbre n'est pas créé avec une interface événementielle.

SAX est une interface événementielle conçue pour le langage Java. Il existe une version pour les langages Python et Perl, mais pas encore pour Javascript ou C++. Elle est publiée par David Meggison, et donc n'émane pas d'un organisme officiel de normalisation comme le DOM. SAX est présente dans le parseur ProjectX de Sun, parseur qui prend aussi en charge le DOM. D'autres parseurs aussi comme celui d'IBM[13] prennent en charge l'interface SAX.
Les événements SAX sont définis comme étant des méthodes attachées à des interfaces Java spécifiques, et ces interfaces sont toutes implémentées dans la classe `HandlerBase`.

Un exemple d'une interface de SAX:
L'interface `parser` : Cette interface définit, par exemple la méthodes `parse()`, pour commencer l'interprétation du document et les méthodes `setDocumentHandler()`, `setDTDHndler()`, `setEntityReslver()`, et `setErrorHandler()` pour enregistrer les événements.

4.1.6 Feuilles de style

Nous avions dit tout au début de ce chapitre que XML ne se préoccupe pas de la présentation des documents, mais s'occupe uniquement du contenu. C'est pour cela que des feuilles de style doivent être utilisées pour faire une présentation particulière du contenu des documents XML.
Les principales feuilles de style utilisées avec des documents XML sont les feuilles CSS et des feuilles XSL.

4.1.6.1 Les feuilles de style CSS

Les feuilles de style CSS (*Cascading Style Sheets*) sont créées en 1996 par le W3C pour ajouter des propriétés de style aux documents HTML de manière standardisée. Il s'agit d'un langage dans lequel on décrit les règles de présentation associées à chaque paramètre. Leur utilisation avec XML est identique à celle faite avec les fichiers HTML, le but étant d'associer à chaque contenu d'une balise un style de présentation.

Après la version de 1996, le W3C a proposé une version révisée et enrichie dénommée CSS niveau 2. C'est d'ailleurs à partir de ce nom que la version de 1996 s'appelle CSS niveau 1. CSS niveau 2 enrichit CSS niveau 1 par la prise en charge des styles audio, des types de média, et bien d'autres fonctions.

Pour ce qui concerne notre travail, avec l'utilisation que nous faisons de XML, les feuilles CSS niveau 1 sont largement suffisantes. C'est pour cela que dans la suite de ce document, à chaque fois que nous parlerons de CSS, nous sous-entendrons CSS niveau 1.

[13] http://www.alphaworks.ibm.com

CSS est partiellement géré par Netscape Navigator 4.0 et par Internet Explorer à 5.0. Les versions suivantes des deux navigateurs ont amélioré leur prise en charge de CSS.

Pour associer une feuille de style à un document XML, il faut insérer dans le document concerné une instruction de traitement qui fait référence à la feuille de style. Si par exemple, notre feuille de style CSS se trouve dans un fichier nommé `style.css`, l'instruction de traitement à écrire à la deuxième ligne du document XML est `<?xml-stylesheet type="text/css" href="style.css"?>`.

Voyons le fonctionnement effectif à l'aide d'un exemple :

Soit la feuille de style suivante dans un fichier nommé `termeglossaire.css` se rapportant aux termes du glossaire simplifié du modèle OWPL.

```
/*  une feuille de style pour les termes du glossaire */
terme {
            display: block;
            margin-bottom: 10px;
            margin : 10 px;
            font-size: 10 pt;
            font-family: Arial;
            padding: 0.2in;
            width : 16 cm

      }
terme intitule {
                font-size : larger;
                font-weight : bold;
                font-style : italic;
                display : block;
                margin-bottom: 10px;
               }
terme signification
                    { display : block;
                      margin: 15pt
                     }
terme source {display:none}
```

La première ligne est un commentaire, avec une syntaxe identique à celle du langage Java. Il y a quatre parties définissant l'affichage associé à chaque balise du document XML. On y définit le mode d'affichage (`bloc`), la marge de bas (`margin-bottom`), la marge (`margin`), la taille des polices de caractère (`font-size`), le type de police (`font-style`), la graisse de police(`font-weight`), la taille du bloc (`width`) ou l'espace par rapport à la bordure(`padding`). Les valeurs de ces différents paramètres sont soit de mots réservés (`larger, bold, none,...`), soit des mesures en pouces (`0.2in`), en centimètres (`16 cm`) ou en pixel (`10px`).

En appliquant cette feuille de style au fichier XML ci-après, on a le résultat sur la figure 5 dans Internet Explorer 5.0.

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/css" href="termeglossaire.css"?>
<termeglossaires>
      <terme>
            <intitule>Codage</intitule>
```

```
            <signification>Traduction de la description technique du
systeme en code operationnel, compte tenu de l'environnement
d'exploitation.[OWPL]</signification>
            <source>OWPL</source>
      </terme>
      <terme>
            <intitule>Acquis</intitule>
            <signification>Experience accumulee dans le passe et
susceptible d'enrichir les projets en cours ou
futurs.[OWPL]</signification>
            <source>OWPL</source>
      </terme>
</termeglossaires>
```

Résultat sur Internet Explorer 5.0:

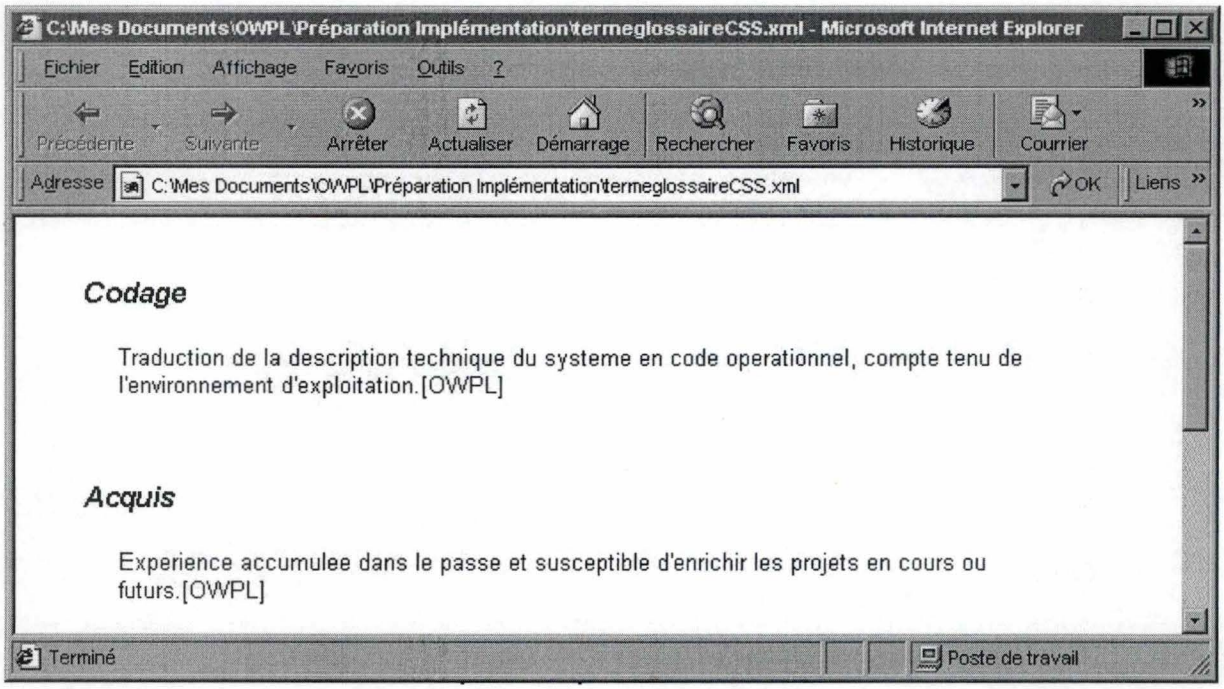


Figure 5: Résultat de l'application d'une feuille de style CSS à un document XML.

4.1.6.2 Les feuilles de style XSL

Les feuilles de style CSS étant plus adaptés au langage HTML, elles ne sont pas suffisantes pour prendre en charge des documents XML. On peut par exemple vouloir trier le contenu d'un fichier XML avant de l'afficher, ajouter des notes de bas de page, des notes de marge, une table des matières, etc. Ce sont toutes des fonctionnalités de XSL qui supplantent les feuilles de style CSS.

Le XSL (*XML StyleSheet Language*) est constitué d'un langage de transformation, XSLT (*XSL Transformation*)et d'un langage de formatage, XSLFO (*XSL Formatting Objects*). Le premier sert à transformer des documents XML en d'autres documents XML suivant un ensemble de règles inscrites dans un fichier XML, tandis que le deuxième se charge de la mise en forme des documents avant affichage. C'est le deuxième, XSLFO, qui est plus proche

de CSS, du moins en ce qui concerne leur but, car XSLFO est beaucoup plus puissant. C'est cette similitude qui fait que plusieurs navigateurs qui implémentent XSL n'implémentent que la partie XSLT, la combinaison XSLT-CSS étant suffisante pour Internet. L'utilisation de XSLFO étant similaire à CSS, voyons comment on peut utiliser XSLT:

Dans la partie qui suit, quand nous parlons de XSL, c'est sous-entendu XSLT.

Par définition, un document HTML respectant la syntaxe XML est un document XML bien formé. C'est cette propriété qui est appliquée pour transformer, à l'aide de XSL, un document en un document HTML pour qu'il soit affichable sur un navigateur. Dans la pratique, on peut associer une feuille de style CSS à ce document pour imposer des règles d'affichage de manière standard.
Le fichier XSL est lui-même un document XML bien formé, qui associe à chaque balise du fichier à transformer, une règle de transformation pour le fichier sortant. Voyons le déroulement avec notre exemple sur le glossaire.

Soit le fichier XSL suivant nommé `termeglossaire.xsl`

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>
       Les termes du glossaire du modele OWPL
</TITLE>
</HEAD>
<BODY>

<xsl:for-each select="termeglossaires">
<xsl:for-each select="terme">
  <P>
          <xsl:for-each select="intitule">
          <B><I> <xsl:value-of />
           </I></B><BR></BR>
          </xsl:for-each>


          <xsl:for-each select="signification">
          <xsl:value-of />
          </xsl:for-each>

          <xsl:for-each select="source">
          [<xsl:value-of/>]
          </xsl:for-each>
  </P>
</xsl:for-each>
</xsl:for-each>

</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

La deuxième ligne (`<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">`), obligatoire, renvoie à l'espace de nom[14] XSL, ceci dans le but d'imposer une seule syntaxe pour les documents XSL. La texte entre les balises `<xsl:for-each select="XXX">` et `</xsl:for-each>` est généré pour dans le fichier cible. Dans ce texte, on peut insérer le contenu de la balise par l'instruction `<xsl:value-of />`.

En appliquant cette feuille de style au fichier suivant, on a le résultat d'affichage sur Internet Explorer 5.0 comme illustré sur la figure 6.

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="termeglossaire.xsl"?>
<termeglossaires>
      <terme>
            <intitule>Codage</intitule>
            <signification>Traduction de la description technique du
systeme en code operationnel, compte tenu de l'environnement
d'exploitation.</signification>
            <source>OWPL</source>
      </terme>
      <terme>
            <intitule>Acquis</intitule>
            <signification>Experience accumulee dans le passe et
susceptible d'enrichir les projets en cours ou futurs.</signification>
            <source>OWPL</source>
      </terme>
</termeglossaires>
```

Résultat sur Internet Explorer 5.0:

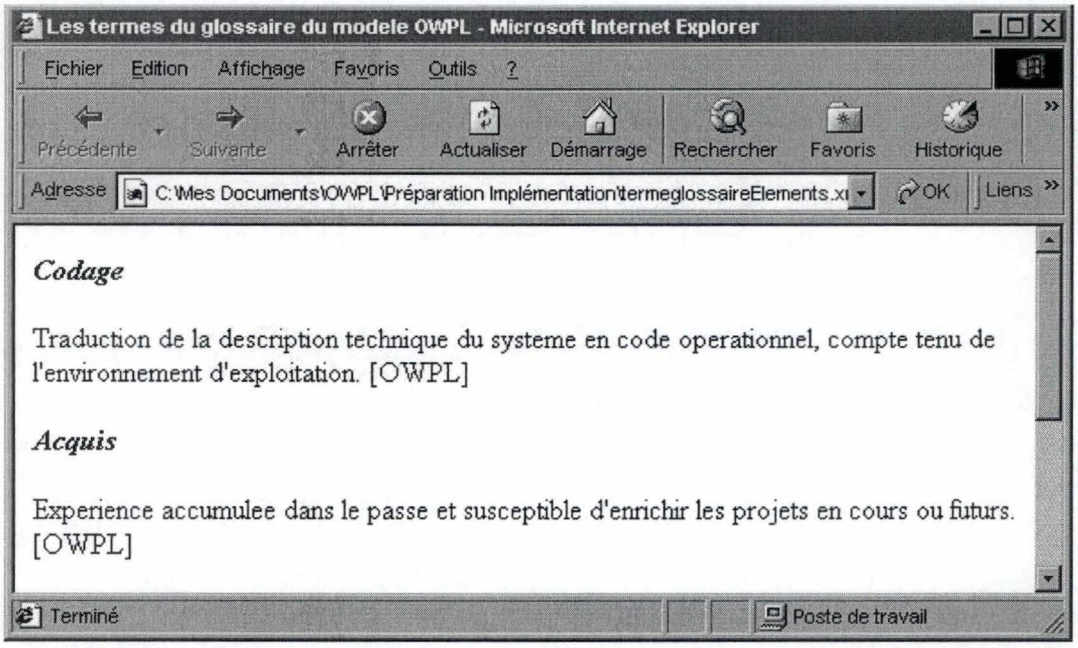


Figure 6: Application d'une feuille de style XSL à un document XML.

[14] Dans le but d'éviter des conflits entre des noms de balises pour des documents XML destinés à être partagés, on peut définir pour une catégorie des fichiers XML, un espace de nom, c'est-à-dire un ensemble de balises à utiliser avec leurs significations. Un URL est alors utilisé pur lier un document XML à son espace de nom.

4.2. Javascript, Applets et Applets signées

4.2.1 Un navigateur comme interface

Le lecteur pourrait se demander l'intérêt de la recherche de notre solution du côté des applications webbased (applications qui tournent dans l'environnement d'Internet, d'Intranet ou Extranet) alors que l'application na paraît pas, à priori, comme une application Internet. La réponse est la suivante : l'application à développer sera plus utilisée pour présenter le modèle et pour le remplissage des questionnaires du modèle OWPL. A ce titre, l'interface d'utilisation, ne fut-ce que dans sa partie présentation doit être sous une forme familière aux utilisateurs. Or, l'utilisation d'Internet s'est généralisé dans les PME Wallonnes, d'où nous avons pensé que l'utilisation presque journalière des navigateurs est assez familière pour que le modèle OWPL soit présenté par leur truchement.

Etant donné également qu'il faut adapter le modèle OWPL avant de l'utiliser dans une entreprise quelconque (En effet, on peut adapter certains vocabulaires propres à l'entreprise pour des pratiques, des délivrables ou des ressources), nous avons recherché une solution des langages qui peuvent permettre de manipuler des documents XML à partir d'une page HTML. C'est pourquoi nous avons cherché du côté de Javascript, des Applets Java premièrement et des Applets Java signées deuxièmement.

4.2.2 Javascript

Javascript est un langage fonctionnant à base d'objets[15] créé dans le but d'étendre le langage HTML. En effet, avec le nombre limité des balises HTML, certaines fonctionnalités sont vites ajoutées avec java script en toute simplicité.

Par rapport à d'autres langages plus structurés, Javascript, étant un langage de script, impose moins de règles et de prescriptions. Par exemple, pour obtenir un affichage, il suffit d'écrire `document.write("Coucou!").`

L'équivalent Pascal serait :

```
Program TestPascal;
Begin
     Writeln("Coucou!");
End.
```

Les équivalents C++ et Java du même programme n'exigent pas moins d'efforts.

[15] La différence entre un langage orienté-objet et un langage fonctionnant à base d 'objets est que le premier permet de définir des objets, les objets du même type étant définis dans des classes, tandis que dans le deuxième, on a un ensemble d'objets préintégrés.

D'après [9], un script est une sorte de langage de programmation permettant de compléter des applications par de petites unités de fonction et de simplifier ou d'automatiser des procédures avec une relative facilité et sans trop de règles ni d'instructions. Cette définition traduit bien l'esprit de Javascript, les langages des scripts dotent les applications d'importantes capacités, très utiles à l'utilisateur.

Sans Javascript, pour apercevoir un document, le serveur effectue tous les calculs nécessaires et le contrôle des formulaires avant de l'envoyer à la machine cliente. Dans le cas des pages enrichies avec du code Javascript, le résultat peut être fourni directement sur la machine cliente, sans être retardé par une nouvelle connexion avec le serveur. Cette information aussi bien être saisie par l'utilisateur (remplissage d'un formulaire) ou figurer dès le départ dans le document HTML enrichi de Javascript. On peut prendre comme exemple ici la publicité dynamique avec utilisation des cookies.

Javascript travaille de manière événementielle. Les événements, très variés, permette à Javascript de réagir. Chaque fois qu'un événement survient, le script adapté à cet événement s'exécute.

Un des avantages de Javascript dans l'application qui nous concerne est qu'il est multi-plateforme à l'exception de quelques fonctions qui ne sont d'ailleurs pas concernées dans le cadre de cette application.

Même si le nom peut être trompeur, Javascript n'a rien à voir avec Java (Javascript n'est ni le sous-ensemble de Java, ni le contraire). D'ailleurs ils ne proviennent pas d'un même éditeur: Javascript a été développé par Netscape et Java par Sun. Le seul point commun, à part la similitude de la syntaxe quelques fois, est la conclusion par les deux firmes des accords concernant la sécurité des deux produits. Leur différence essentielle se situe également dans le but: Java est un langage de programmation complet et Javascript est un langage script destiné à compléter des documents HTML.

Le tableau suivant tiré de [9] montre les différences essentielles entre Javascript et Java:

Javascript	Java
Est interprété par le client.	Est compilé par le programmeur avant d'être exécuté par le client.
Le code est intégré en HTML	Le code (applet) est chargé en tant que module.
Utilisation peu compliquée des données/types de données.	Règles très strictes pour les types de données (doivent être déclarées).
Code utilisable uniquement dans le document HTML.	Permet de créer des programmes autonomes.
Collabore avec des éléments HTML.	Dépasse les possibilités HTML.
Permet un accès direct aux objets navigateur.	Pas d'accès aux objets et fonctionnalités du navigateur.
Fonctionnant à base d'objets: on peut utiliser des objets intégrés, mais il n'y a ni formation de classes ni héritage.	Orienté Objet: les applets se composent de classes avec héritage.
Liaison dynamique: les références objet sont examinées pendant l'exécution.	Liaison statique: les références objet doivent être définies lors de la compilation.

4.2.2.1 Qu'en est –il de Javascript et XML?

Le DOM est implémenté dans certains navigateurs, notamment Internet Explorer 5.0. Tous les exemples qui suivent concernant la manipulation des documents XML via le DOM ont été testés sur Internet Explorer 5.0.

Dans IE 5.0, on utilise le mécanisme d'*îlot XML* pour manipuler du XML. Cette balise est propre à Internet Explorer. Un îlot XML s'écrit comme suit dans du code HTML destiné au navigateur IE 5.0:

```
<xml id="nomIlot"></xml>
```

Avec Javascript, on peut utiliser le DOM ou s'en passer pour manipuler des fichiers XML. En effet, si on veut utiliser le DOM, il suffit d'utiliser l'API DOM. Pour se passer du DOM, on passe par une manipulation simple de texte pour former un document respectant la syntaxe XML qu'on peut transformer ensuite en document XML par l'utilisation de l'îlot XML. Nous allons voir les deux façons de faire avec l'exemple suivant:

Enoncé de l'exemple: Il s'agit de la manipulation des éléments dans un document XML. Nous avons pris comme structure du document, une structure assez simplifiée des termes du glossaire du modèle OWPL. Pour rappel, chaque terme est constitué d'un intitulé qui est son identifiant, d'une signification et d'une source.

4.2.2.2 Exemple d'utilisation du DOM

Soit nous voulons ajouter un terme à notre glossaire. Nous disposons dans notre page HTML d'un formulaire nommé "`form`",avec des zones de saisies de texte nommées "`intit`", "`signif`", et "`srce`", où sont insérées les informations concernant un nouveau terme. Nous y disposons aussi d'un îlot XML nommé "`xmldocument`".

Soit le document XML a la forme suivante :

```
<?xml version="1.0" standalone="yes"?>
<termeglossaires>
      <terme>
            <intitule>Codage</intitule>
            <signification>Traduction de la description technique du
système en code opérationnel, compte tenu de l'environnement
d'exploitation.[OWPL]</signification>
            <source>OWPL</source>
      </terme>
      <terme>
            <intitule>Acquis</intitule>
            <signification>Expérience accumulée dans le passe et
susceptible d'enrichir les projets en cours ou
futurs.[OWPL]</signification>
            <source>OWPL</source>
      </terme>
</termeglossaires>
```

Le code Javascript suivant suffit pour ajouter ce terme dans le document.:

```
var intitule = form.intit.value,
    Signfication = form.signif.value,
    Source = form.signif.value;
```

Pour récupérer les données sur le formulaire

```
Var topLeVel = xmldocument.documentElement;
```

Pour récupérer l'élément de plus haut niveau du document XML, ici l'élément "termeglossaires".

```
var elementTerme = xmldocument.createElement("terme");
var elementIntitule = xmldocument.createElement("intitule");
var elementSignification = xmldocument.createElement("signification");
var elementSource = xmldocument.createElement("source");
```

Ici, nous avons déjà formé des éléments vides avec comme balises "terme", "intitule", "signification" et "source".

```
var text = xmldocument.createTextNode(intitule);
elementIntitule.appendChild(text);

text =  xmldocument.createTextNode(signification);
elementSignification.appendChild(text);

text = xmldocument.createTextNode(source);
elementSource.appendChild(text);
```

Nous avons ici ajouté les textes saisis dans le formulaire pour l'intitulé, la signification et la source. Il nous reste maintenant à créer un élément Terme constitué des trois éléments correspondants (intitule, signification et source), puis le rattacher au à l'élément de plus haut niveau du document XML. C'est ce qui est fait avec le code qui suit

```
elementTerme.appendChild(elementIntitule);
elementTerme.appendChild(elementSignification);
elementTerme.appendChild(elementSource);
topLevel.appendChild(elementTerme);
```

A la fin ici, nous avons un nouveau document constitué de celui qui était avant dan l'îlot XML auquel on a ajouté un nouveau terme. Tout ceci est fait grâce à l'API DOM qui fournit toutes méthodes relatives à différents types d'éléments.

4.2.2.3 Exemple d'utilisation sans le DOM

Contrairement au point précédent, on peut manipuler un document XML en se passant de l'API DOM telle qu'implémentée dans Internet Explorer 5.0. L'avantage est ici qu'on a un code Javascript qui manipule du simple texte, et donc susceptible de s'exécuter sur un autre navigateur. Le seul problème c'est quand il faut transformer le code XML généré en document

XML. Ici nous utilisons un îlot XML pour ce faire. Pour pallier à ce problème dans un autre navigateur, la seule solution si ce navigateur n'implémente pas le DOM, c'est de pouvoir écrire sur un fichier le code XML généré. Ce qui n'est pas possible comme nous le verrons à la fin de ce chapitre.

Nous avons utilisé trois listings que nous allons expliquer au fur et à mesure: un fichier avec le code Javascript, un fichier avec le cde HTML de la page, et un fichier avec la feuilles de style XSL du document XML. Nous verrons plus-tard pourquoi il n'y a pas de fichier XML.

Soit le document XML est défini par la DTD suivante :

```
<!DOCTYPE termeglossaires [
<!ELEMENT termeglossaires(terme*)>
<!ELEMENT terme(intitule, signification, source)>
<!ELEMENT intitule(#PCDATA)>
<!ELEMENT signification(#PCDATA)>
<!ELEMENT source(#PCDATA)>
]>
```

Le code de la feuille de style XSL :

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>
      Les termes du glossaire du modele OWPL
</TITLE>
</HEAD>
<BODY>

<xsl:for-each select="termeglossaires">
<xsl:for-each select="terme">
  <P>
         <xsl:for-each select="intitule">
         <B><I> <xsl:value-of />
          </I></B><BR></BR>
          </xsl:for-each>


         <xsl:for-each select="signification">
         <xsl:value-of />
         </xsl:for-each>

         <xsl:for-each select="source">
         [<xsl:value-of/>]
         </xsl:for-each>
  </P>
</xsl:for-each>
</xsl:for-each>

</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

Cette feuille de style est utilisée pour transformer le document XML en un document HTML avec un style défini par nous mêmes. Par exemple, les intitulés seront gras et en italique, et les sources seront entre crochets.

Le code HTML de la page :

```
<HTML>
<HEAD>
  <META NAME="GENERATOR" CONTENT="Adobe PageMill 3.0 Win">
  <TITLE>Traitement des terme du glossaire (modele OWPL)</TITLE>
  <SCRIPT LANGUAGE="JavaScript" SRC="createlistFinal.js"></SCRIPT>
</HEAD>
<BODY BGCOLOR="#ffffff">

<FORM ID="controls">
<P><CENTER><TABLE WIDTH=70% BORDER="0" CELLSPACING="2" CELLPADDING="0"
HEIGHT="121">
  <TR>
    <TD WIDTH="100%">
    <P><CENTER> <FONT COLOR="#0000ff" SIZE="+1">Terme</FONT><B><FONT
     SIZE="+1"> </FONT></B></CENTER></P>

    <P><CENTER><B></B><TABLE WIDTH="516" BORDER="0" CELLSPACING="2"
    CELLPADDING="0">
      <TR>
        <TD WIDTH="50%">
         Intitule:</TD>
        <TD WIDTH="50%">
         <INPUT NAME="intitule" TYPE="text" SIZE="25"></TD>
      </TR>
      <TR>
        <TD WIDTH="50%">
         Signification :<B> </B></TD>
        <TD WIDTH="50%">
         <INPUT NAME="signification" TYPE="text" SIZE="50"></TD>
      </TR>
      <TR>
        <TD WIDTH="50%">
         Source : </TD>
        <TD WIDTH="50%">
         <INPUT NAME="source" TYPE="text" SIZE="25"></TD>
      </TR>
    </TABLE></CENTER></P>

    <P><CENTER>

              <INPUT  NAME="BoutonAjouter" TYPE="BUTTON" VALUE="Ajouter"
                ONCLICK="addTerme(controls)" >     

               <INPUT  TYPE="BUTTON" VALUE="Supprimer"
                ONCLICK="deleteTerme(controls)"> 


              <SELECT NAME="termeList" SIZE="1" WIDTH="10"></SELECT> 

              <INPUT  TYPE="BUTTON" VALUE="Editer"
                ONCLICK="EditTerme(controls)"> 


              <INPUT  TYPE="BUTTON" VALUE="Version XML"
                ONCLICK="exportTerme(controls,xml,xslt)"> 

                <INPUT  TYPE="BUTTON" VALUE="Version HTML"
                ONCLICK="exportTermeHTML(controls,xml,xslt)"> 



  </TR>
</TABLE>

              <TEXTAREA NAME="output" ROWS="22" COLS="70" READONLY>
              </TEXTAREA>

</FORM>
<xml id="xml"></xml>
<xml id="xslt" src="termeglossaireElements.xsl" ></xml>
</CENTER>
</BODY>
</HTML>
```

Sur cette page HTML, nous définissons un formulaire avec tous les boutons de manipulation des terme du glossaire. Il s'agit des boutons "Ajouter", "Supprimer", "Editer", "Version XML", et "Version HTML". A chaque bouton est associée une action qui est décrire dans le fichier Javascript nommé "`createlistFinal.js` " inséré avec le code suivant : `<SCRIPT LANGUAGE="JavaScript" SRC="createlistFinal.js"></SCRIPT>`.

Vers la fin du document, deux îlots XML sont utilisés pour utiliser le parseur DOM de Internet Explorer 5.0 et la feuille de style définie plus haut.
Il s'agit du code:

```
<xml id="xml"></xml>
<xml id="xslt" src="termeglossaireElements.xsl" ></xml>
```

Le **code Javascript** correspondant à cette partie se trouve en annexe. Il s'agit du document n°3.

Toutes les fonctions déclarées dans ce fichier sont utilisées lors de la survenance d'un clic sur un bouton sur le fichier HTML. Pour les comprendre, voyons ce qui se passe lorsqu'on appuie sur le bouton "Ajouter".

Dans ce cas, la fonction `addTerme` est appelée avec comme argument, le formulaire `controls`. Cette fonction commence par vérifier que le terme qu'on veut ajouter n'existe pas déjà dans le glossaire, ceci dans le but de respecter la contrainte d'unicité de l'identifiant.
Si le terme n'existe pas encore, l'intitulé, la signification et la source sont récupérés, et un nouveau terme est ajouté sur le tableau `termes`, la liste des termes. Ensuite on fait appel à la fonction `exportTerme` avec comme arguments le formulaire `controls` et les îlots XML `xml` et `xslt`. En appelant d'autres fonctions, cette fonction forme du texte XML syntaxiquement correcte correspondant au terme qu'on veut ajouter, et effectue une mise à jour du code XML de l'ensemble du document.

Lorsqu'on appuie sur les boutons "Supprimer" et "Editer", on utilise Javascript de façon classique pour enlever un élément dans un tableau ou le modifier.

L'utilisation des boutons "version XML" et "Version HTML a deux réactions différentes: pour le premier, il s'agit d'un simple appel de la fonction Javascript `makeXML()` qui transforme le contenu du tableau des termes en un texte représentant du code XML pour les termes du glossaire. Pour le second par contre, on appelle la fonction `exportTermeHTML` qui prend comme arguments le formulaire `form` et les deux îlots `xml` qu'on a nommé `xml` et `xslt`. Cette dernière fonction utilise le DOM pour transformer le texte que nous manipulons en document XML à l'aide de la commande `xml.loadXML(xmlDOC)`, avec `xmlDOC` le texte représentant du code XML. La méthode `loadXML` affecte à l'îlot XML auquel elle s'applique le document XML représenté par le texte qu'elle reçoit en argument. Enfin, pour appliquer la feuille de style au document de l'îlot XML, on a appliqué la méthode `transformNode(<feuille de style>)` qui renvoie du texte représentant le fichier XML auquel on a appliqué la feuille de style donnée en argument.

Résultat sur Internet Explorer 5.0 après ajout de deux éléments:

Vue de la version XML du document

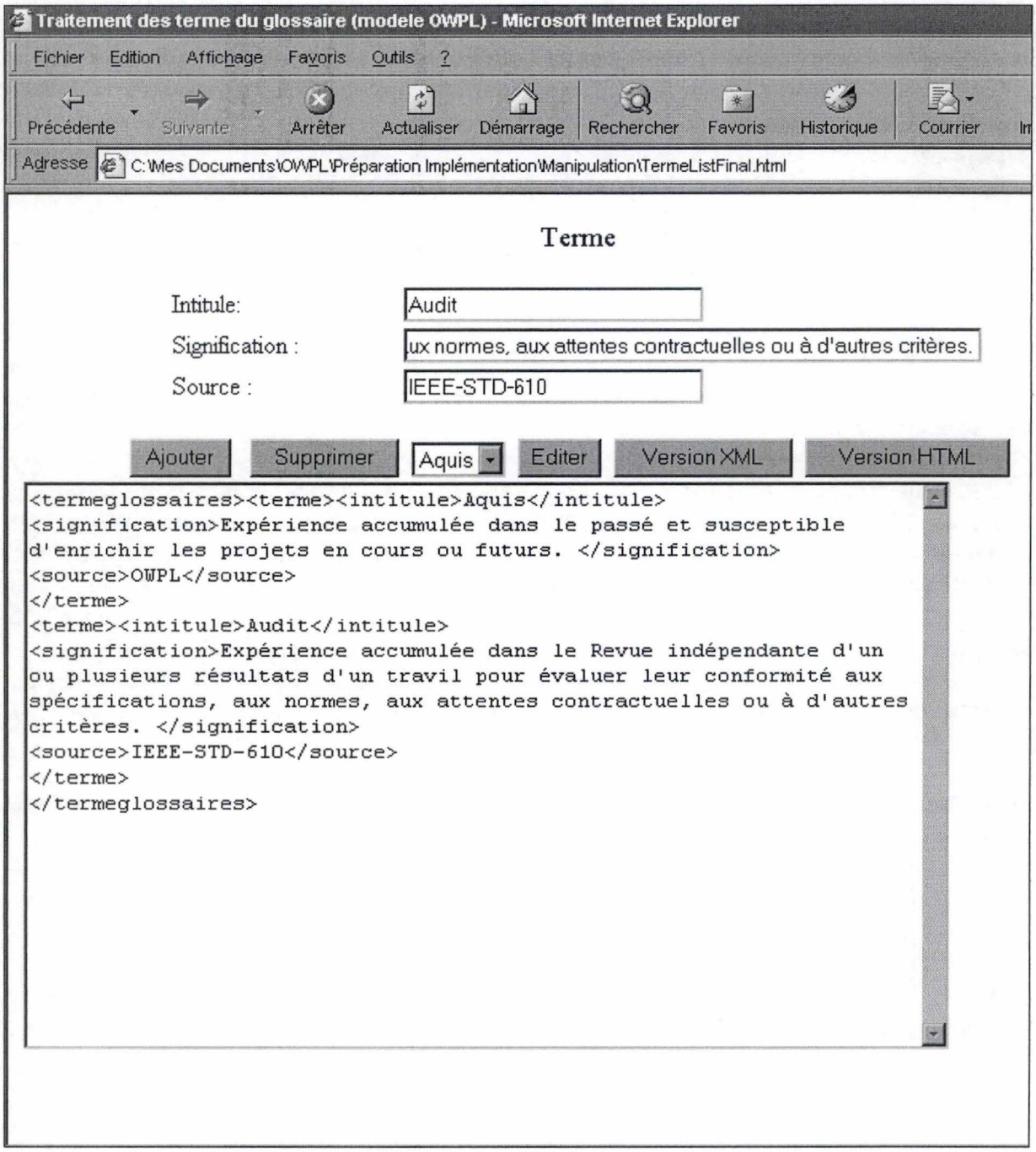


Figure 7 Aperçu de la manipulation d'un document XML sur Internet Explorer 5.0

On a sur la figure 7 ci-dessus l'aperçu de la version XML sur le navigateur Internet Explorer 5.0 du code HTML donné plus haut. En cliquant sur le bouton "version HTML", nous appliquons la feuille de style XSL donnée plus haut à ce document XML, et le résultat est donné par la figure 5.

Vue de la version HTML du document

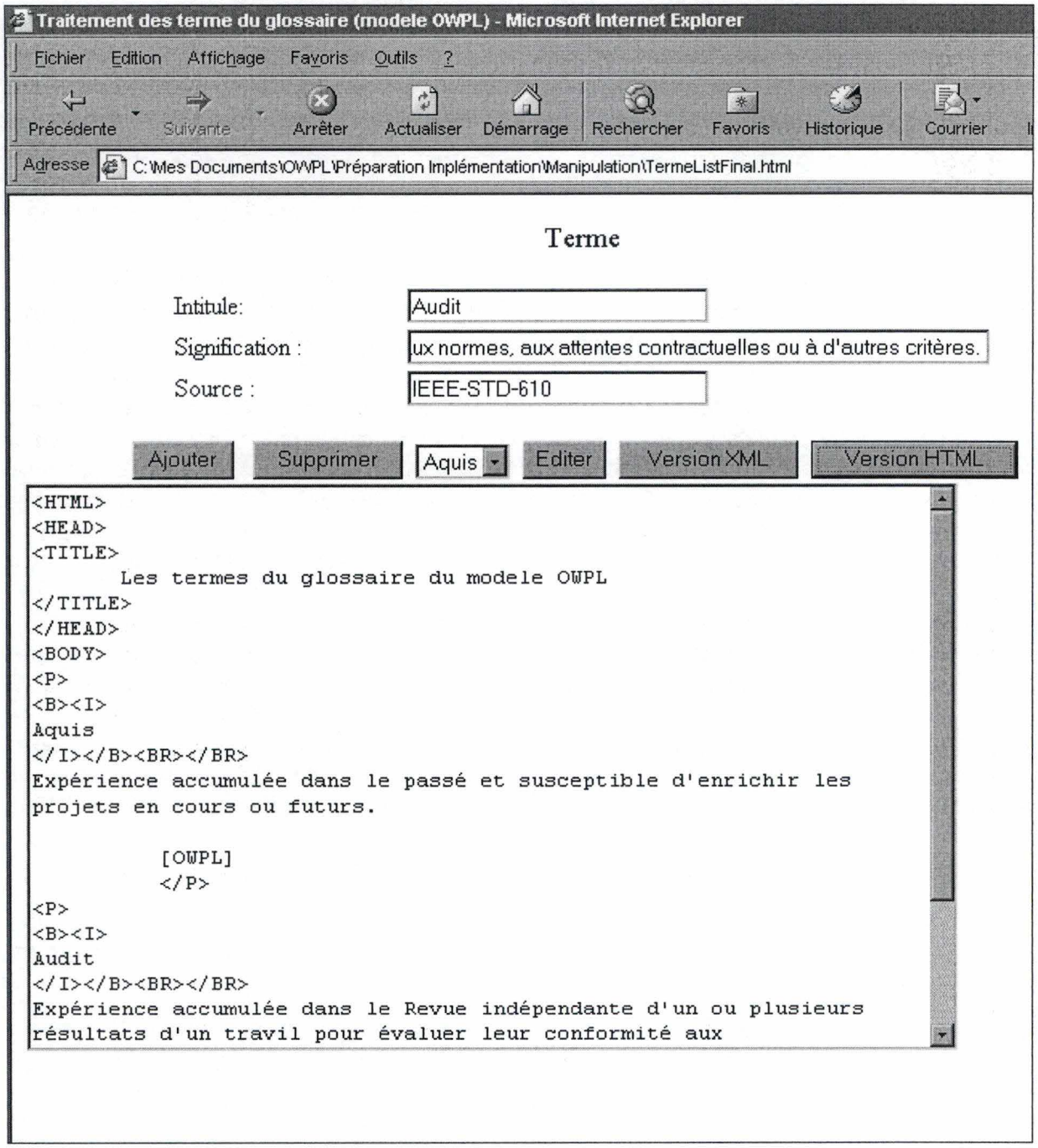


Figure 8 Aperçu de la manipulation d'un document XML sur Internet Explorer 5.0 avec utilisation d'une feuille de style XSL.

4.2.2.4 Sauvegarde des fichiers avec JavaScript

Dans tous les points qui précèdent, nous avons vu comment manipuler XML avec Javascript. Cette manipulation est facilitée avec l'API DOM implémentée dans Internet Explorer 5.0, mais, comme nous l'avons dit au début de ce travail, l'application que nous voulons concevoir doit être portable, d'où il serait aberrent de retenir cette solution pour notre application.

Cette dernière raison n'est d'ailleurs pas la seule dans la mesure où, nous avons vu qu'il est possible de manipuler des documents XML sans utiliser l'API DOM, à condition de pouvoir sauver les textes que nous manipulons sous forme des fichiers XML. C'est là qu'il y a un autre problème qui écarte l'utilisation de Javascript pour notre application.

En effet, le but premier de Javascript est d'étendre les possibilités du HTML. Cette extension se fait moyennant des règles strictes de sécurité sur Internet. C'est au nom de cette sécurité que les applications Javascript ne peuvent accéder au disque local de la machine cliente. Les fichiers souvent utilisées avec Javascript sont les Cookies[16], mais les cookies sont gérés par le navigateur, et aucun code Javascript ne les manipule directement.

4.2.3 Applets Java

Pour cette partie, la connaissance du langage Java est un pré-requis. Le sujet sur le langage Java étant très large, nous ne pouvons nous y étendre sans nous écarter sensiblement du présent travail. Le lecteur intéressé peut trouver une bonne présentation du langage Java dans la référence bibliographique [19].

Du code Java peut être exécuté sur une page HTML. Pour ce faire, il faut créer une catégorie spécifique d'objet : les applets Java. Une applet Java est donc un programme Java qui tourne sur un navigateur par le canal d'une page HTML.

Les applets utilisent une interface fenêtrée affichée dans une zone rectangulaire d'une page HTML et leur contexte d'exécution est fourni par le navigateur qui affiche la page HTML. C'est pour cela qu'il faut s'assurer qu'un navigateur est compatible Java avant d'y exécuter une applet. Suivant qu'on veut utiliser AWT ou SWING[17], une applet est une classe qui étend la classe `java.awt.Applet` ou la classe `javax.swing.JApplet`.

Exemple d'une petite appet Java.

Le code Java :

```
import javax.swing.*;
public class PremiereApplet extends JApplet {
public void init() {
```

[16] Les cookies représentent un mécanisme universel capable d'utiliser la connexion à un serveur pour sauvegarder et rappeler des données du côté client. Cette possibilité de sauvegarder en permanence l'état du client étend considérablement les capacités des applications client/serveur Web[9].

[17] AWT et SWING sont des classes Java qui implémentent l'interface graphique.

```
getContentPane().add(new JLabel("Ça marche !"));
}
}
```

De manière générale, on peut dire que la différence avec une application Java se situe dans le fait que le lancement d'une applet se fait par l'exécution de la méthode `init()`, au lieu de la méthode `main()`.

Le code de la page HTML qui l'utilise :

```
<html>
<body>
<applet code="PremiereApplet" width="200" height="150">
</applet>
</body>
</html>
```

L'insertion d'une applet dans une page HTML passe par la balise `<applet>`, avec comme caractéristiques le nom du fichier comportant l'applet (`code="PremiereApplet"`) et la taille de la fenêtre où va s'exécuter l'applet (`width="200" height="150"`).
La troisième ligne de ce code HTML signifie ceci: le code Java a été compilé et se trouve dans un fichier appelé `PremierApplet.class`.

A part quelques problèmes dus à des versions de Java dans les navigateurs, tout code Java est susceptible de s'exécuter au travers d'une applet sur un navigateur. On peut donc dire que nous sommes en face d'une technologie favorable à notre application: non seulement c'est portable, on a en plus la possibilité de manipuler du XML à l'aide d'un parseur Java, puisque le code correspondant à l'application peut être enveloppée dans une applet Java. Il y a malheureusement une seule restriction: pour des raisons de sécurité, le code Java d'une applet de peut manipuler des fichiers logés sur la machine cliente. En effet, on pourrait alors facilement écrire des applets qui peuvent manipuler le système des fichiers de la machine cliente, avec possibilité de transmission des virus informatiques. C'est pour pallier à ce problème qu'à partir de Java 2, une applet doit être signée avant d'accéder à des fichiers sur la machine cliente : on parle alors d'applets signées.

4.2.4 Applets signées

Restreindre l'accès aux fichiers locaux à une applet dans un environnement sûr comme un intranet, où un site Internet sûr auquel on est abonné est trop sévère. C'est pour cela qu'un utilisateur peut accorder cette permission à des applets d'origine sure et connue. Cette démarche suppose deux choses : L'utilisateur doit avoir un mécanisme sûr de reconnaissance des applets et les développeurs doivent avoir un mécanisme sûr de signature d'applets.

Du côté des utilisateurs, Java 2 permet de résoudre ce problème en configurant le *security manager* [18]afin qu'ils puissent autoriser les applets d'origine connue à accéder au disque dur local. A travers des fichiers sur lesquels sont inscrites les permissions, on définit des applets à des actions permises sur le système client.

[18] Le *Secutity Manager* est un programme permettant de gérer tous les fichiers de permissions accordées à des applets ou des applications Java.

Du côté des développeurs, Java 2 offre également un mécanisme de signature des applets. C'est une méthode basée sur la cryptographie à clé publique[19].

La cryptographie à clé publique est fondée sur la notion de clé publique et clé privée. L'idée est de garder secrète la clé privée et de publier la clé publique. A partir des propriétés des deux clés qui sont liées mathématiquement, une fichier crypté avec la clé publique ne peut être décrypté qu'avec la clé publique. Inversement, à partir de la clé publique, on sait vérifier si un fichier a été signé avec la clé privée correspondante. Dans ce cas, ce que l'on veut, ce n'est pas crypter un texte mais certifier l'auteur de celui-ci.

C'est ainsi que le développeur d'une applet doit utiliser sa clé privée pour signer son applet, et après avoir envoyé sa clé publique aux gens qui vont utiliser son code, ces derniers peuvent bien vérifier que cette applet été signée par lui avant d' accorder à l'applet la permission d'accéder à leurs fichiers locaux en utilisant le *Security Manager*. Une explication plus détaillée concernant le mécanisme de signature d'applet est fournie avec le document N° 5 en annexe.

Une chose cependant à signaler ici est qu'à chaque navigateur est associé un mécanisme de permission pour les applets signées: ce qui est valable pur Internet Explorer n'est pas forcément valable pour le navigateur HotJAva ou Netscape Navigator.

Pour ce qui est de notre application, il semble qu'avec l'utilisation d'applets signées, nous pouvons écrire un programme Java (une applet) qui puisse tourner dans un navigateur. Par ailleurs, ce programme va pouvoir utiliser un parseur java pur manipuler la base des données constituée des fichiers XML. Voyons dans le point suivant ce que nous avons retenu comme solution pour notre application, avec les arguments qui ont déterminé notre choix.

4.3. Solution finale : Application Java, fichiers XML et HTML

4.3.1 Pourquoi cette solution?

Nous avons vu tout au long de ce chapitre plusieurs technologies qu'on peut utiliser pour arriver à manipuler des fichiers XML. Notre choix dépendait de l'exigence principale de notre application: **la portabilité**.

Javascript offre les possibilités nécessaires à notre application, mais son problème majeur reste le fait qu'on ne peut accéder à des fichiers sur la machine cliente. Un autre désavantage reste le fait que peut de navigateur actuellement offrent des parseurs XML intégrés. Adopter l'utilisation de Javascript, même si on aurait pu accéder à des fichiers sur la machine cliente, signifierait que notre application dépend de Internet Explorer (à partir de 4.0) ou l'une des toutes dernière version de Netscape Navigator (à partir de la version 6.0).

Une autre opportunité s'est offerte avec les applets Java, mais, comme on a vu, la restriction sur les accès aux fichiers est ici aussi un gros désavantage. La solution des applets signées paraît séduisante, mais elle est fort limitée : Chaque navigateur a un système unique

[19] Une connaissance détaillée de la cryptographie à clé publique peut être obtenue dans la référence[19]

de gestion des permissions pour les applets signées. On peut se dire qu'il suffirait alors d'appliquer la méthode adéquate dans une entreprise évaluée avec le modèle OWPL. Non seulement c'est susciter la susceptibilité de ces entreprises au niveau de leurs mesures de sécurité, mais adopter une telle démarche suppose qu'il faut maîtriser les techniques de gestion d'applets signées pour tous les navigateurs potentiels utilisables dans les PME Wallonnes. C'est un problème qui dépasse le cadre de ce travail, car il nécessite une investigation poussée avant d'adopter une solution quelconque.

Il nous reste donc à tirer les avantages qu'on avait en utilisant les applets Java: l'utilisation du langage Java multi-plateformes et l'utilisation des navigateurs pour visualiser le modèle OWPL. C'est pourquoi la solution finale consiste en une application Java qui, par nature du langage, est multi-plateformes. Avec l'application Java, on va se servir des fichiers XML comme base des données. Ces fichiers XML serviront uniquement comme de base des données, et le programme se chargera de générer, un ensemble de documents HTML, qui présenteront le modèle OWPL.

4.3.2 Utilisation concrète

4.3.2.1 Parseur utilisé

Ce n'est sans doute pas par hasard que notre choix s'est porté sur le langage Java : il se fait que le langage Java est privilégié pour le développement XML. La plupart des outils XML qui existent sont écrits en Java. Il existe probablement plus de parseurs XML gratuits écrits en Java que des parseurs écrits dans tous les autres langages. Nous n'avions donc que l'embarras du choix quant au parseur à utiliser.

Parmi les parseurs les plus utilisées, il y a le parseur *IBM for Java* de chez IBM, le parseur de Microsoft, le parseur de Oracle, et le parseur *ProjectX* de chez Sun. Les parseurs IBM for Java et ProjectX prennent en charge aussi bien l'interface DOM que l'interface SAX. Etant donné que notre application n'utilise pas XML de manière très particulière et poussée, le choix de l'un ou l'autre de ces deux parseurs n'avait aucune importance : nous avons opté pour le parseur ProjectX.

Nous utilisons l'interface DOM de ce parseur pour exploiter les fichiers XML qui constituent la base des données de l'application. Pour rappel, utiliser une interface DOM d'un parseur avec un fichier XML revient à créer un arbre respectant la recommandation DOM à partir des données d'un fichier XML. C'est en exploitant les nœuds de cet arbre qu'on arrive à reconstituer notre le modèle OWPL dans l'application.

4.3.2.2 Structure des documents XML

Nous utilisons des documents XML ici pour sauvegarder le modèle OWPL, dans un ensemble des fichiers texte (donc multi-plateformes). Un problème à résoudre est de savoir si nous allons utiliser des éléments ou des attributs pour nos fichiers XML.

En effet, on a le choix entre les éléments et les attributs pour représenter les données dans un fichier XML. Par exemple, pour notre glossaire vu plus haut, on a le choix entre la structure du fichier 1 et celle du fichier 2.

Fichier 1 qui utilise des éléments :

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="termeglossaire.xsl"?>
<termeglossaires>
      <terme>
            <intitule>Codage</intitule>
            <signification>Traduction de la description technique du
systeme en code operationnel, compte tenu de l'environnement
d'exploitation.</signification>
            <source>OWPL</source>
      </terme>
</termeglossaires>
```

Fichier 2 qui utilise des attrributs :

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="termeglossaire.xsl"?>
<termeglossaires>
      <terme
            intitule = "Codage"
            signification = "Traduction de la description technique du
                        systeme en code operationnel, compte tenu de
                        l'environnement d'exploitation."
            Source = "OWPL"
       />
</termeglossaires>
```

La manipulation des deux structures avec notre parseur se fait à des niveaux de difficultés égaux. Néanmoins, l'utilisation des attributs se fait de manière générale pour la forme des données (informations qui déterminent leur présentation par exemple) et les éléments sont utilisés pour le contenu. Nous avons donc opté pour les éléments, par qu'ils donnent une meilleure idée de la structure de notre base des données et offrent des meilleurs possibilités d'extension ou de réutilisation éventuelles.

Une fois ce choix déterminé, on a le choix entre faire une seul fichier hiérarchisé ou plusieurs fichiers XML. Dans le cas d'un seul fichier XML, il s'agit du seul fichier qui représenterait le modèle OWPL, et dans le cas de plusieurs fichiers, il s'agit des fichiers XML qui correspondraient chacune à une table d'une base de données relationnelles.

Dans le cas d'un seul fichier hiérarchisé, on peut aboutir à un fichier comme celui ci-dessous pour le modèle OWPL:

```
<modeleOWPL>
<processus>
      <nom><nom/>
      <reference></reference >
      <objectif></objectif>
      <description></description >
      <pratiques-essentielles>
      <pratique>
            <nom></nom>
```

```
            <reference></reference>
            <objectif></objectif>
            <entrees></entrees>
            ...
            <sorties></sorties>
            ...
            <ressources></ressources>
            ...
            <poids></poids>
      </pratique>
      <pratique>
      ...
      </pratique>
      </pratiques-essentielles>
<processus>
<processus>
...
<processus>
</modeleOWPL>
```

Le désavantage avec une telle structure, à part la fait qu'elle est lourde à gérer, est le fait qu'elle n'est pas favorable à l'évolution de la structure du modèle OWPL, elle n'offre pas un bon aperçu général de la structure du modèle OWPL, et elle permet une redondance des informations[20].

Notre choix s'est porté donc sur plusieurs fichiers XML. Pour chaque table du schéma relationnel de notre base des données, nous avons créé un fichier XML de même nom. **Nous n'avons pas ici la prétention de simuler un système de gestion de base des données relationnelles**, mais nous estimons qu'une telle approche facilite une évolution de la structure du modèle, et permet de mieux mettre en évidence la relation qu'il y a entre les différents objets du modèle.

Nous avons aussi opté pour une non validation des documents par rapport à leurs DTD, car la validation est plus adaptée à un échange des fichiers XML dont on a fixé la structure. Ici, les fichiers XML sont écrits de manière transparente pour l'utilisateur, il n'a aucune emprise sur l'écriture des fichiers, et donc il n'y a aucun risque que le fichier écrit ne respecte plus la spécification de départ.

Notre base des données est finalement composée d'autant des fichiers XML qu'il y a des tables dans le schéma relationnel de la base des données. Tous les fichiers XML sont de la forme :

```
<nom-table+"s">
      <nom-table>
      <colonne-1></colonne-1>
      <colonne-2></colonne-2>
      ...
      <colonne-n></colonne-n>
      </nom-table>
<nom-table+"s" >
```

[20] Si, par exemple, une ressource est utilisée par plusieurs pratiques, elle doit être écrite dans la description de chaque pratique pour laquelle elle est utilisée.

<nom-table+"s"> signifie que pour chaque fichier XML, l'élément de plus haut niveau est entre les balises constituées part le nom de la table sur le schéma conceptuel auquel on a ajouté un "s" à la fin. Pour chaque enregistrement présent dans le fichier, on ajoute un élément <nom-table></nom-table>. Il y a donc autant d'éléments <nom-table> qu'il y a d'enregistrements sur la table correspondant au fichier XML. Les éléments <colonne-n></colonne-n> représentent les champs correspondants aux enregistrements.
Par exemple, pour la table terme-glossaire du schéma relationnel, on a le fichier XML suivant:

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="termeglossaire.xsl"?>
<termeglossaires>
      <terme>
            <intitule>Codage</intitule>
            <signification>Traduction de la description technique du
systeme en code operationnel, compte tenu de l'environnement
d'exploitation.</signification>
            <source>OWPL</source>
      </terme>
      <terme>
            <intitule>Acquis</intitule>
            <signification>Experience accumulee dans le passe et
susceptible d'enrichir les projets en cours ou futurs.</signification>
            <source>OWPL</source>
      </terme>
</termeglossaires>
```

Quelques exemples de fichiers XML utilisés dans notre application sont donnés en annexe au document n°5.

4.3.2.3 Schéma relationnel de la base des données

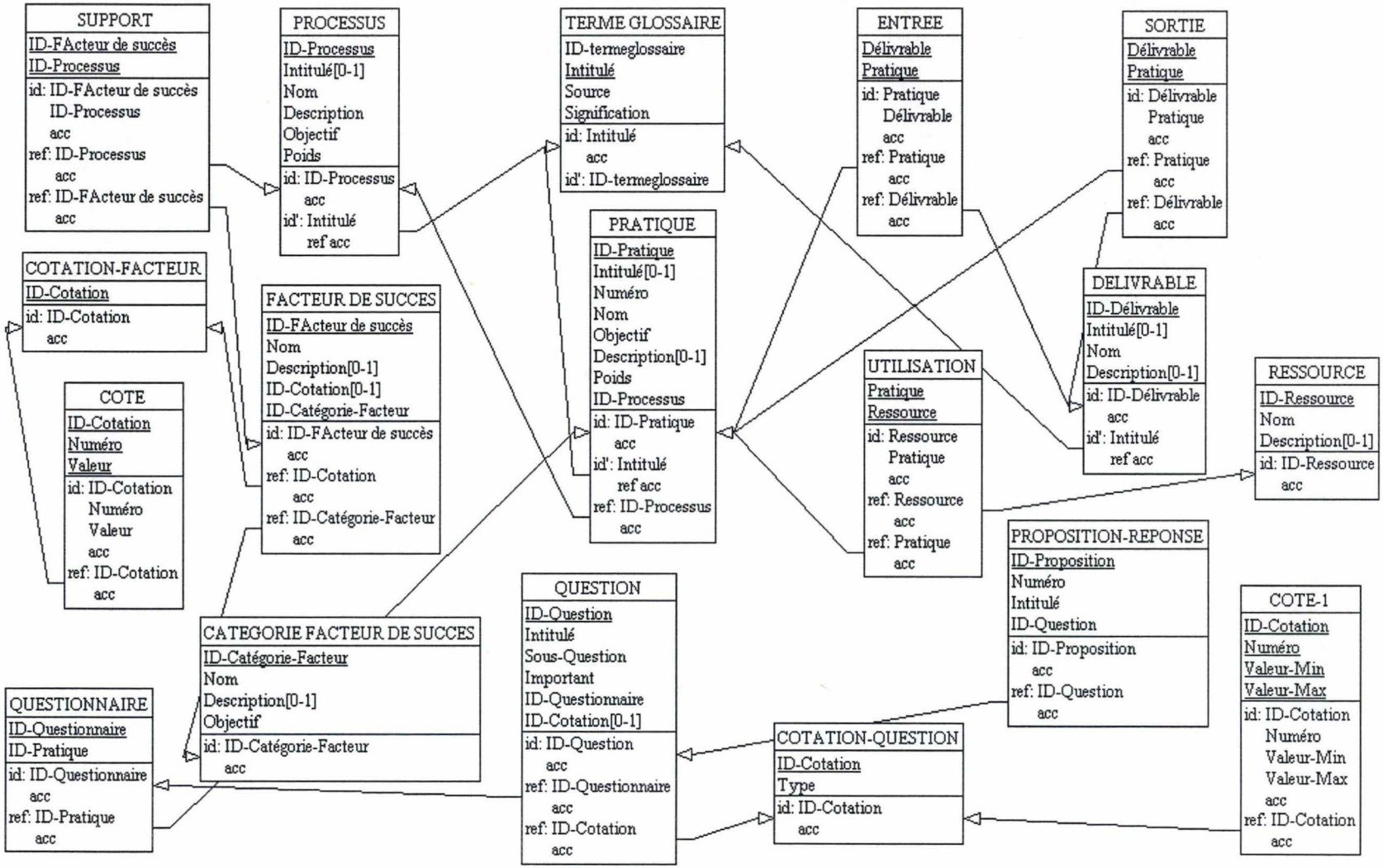
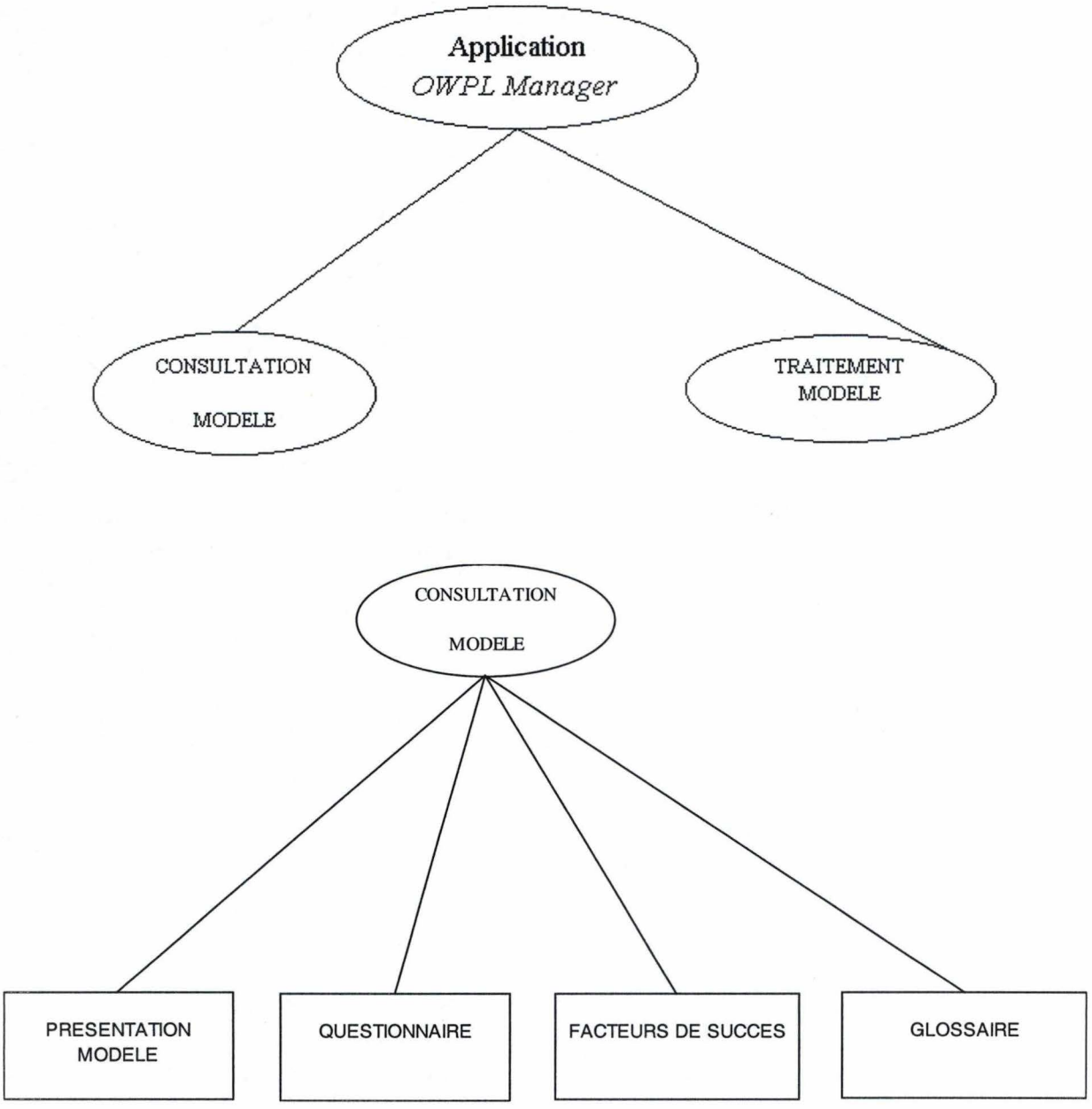


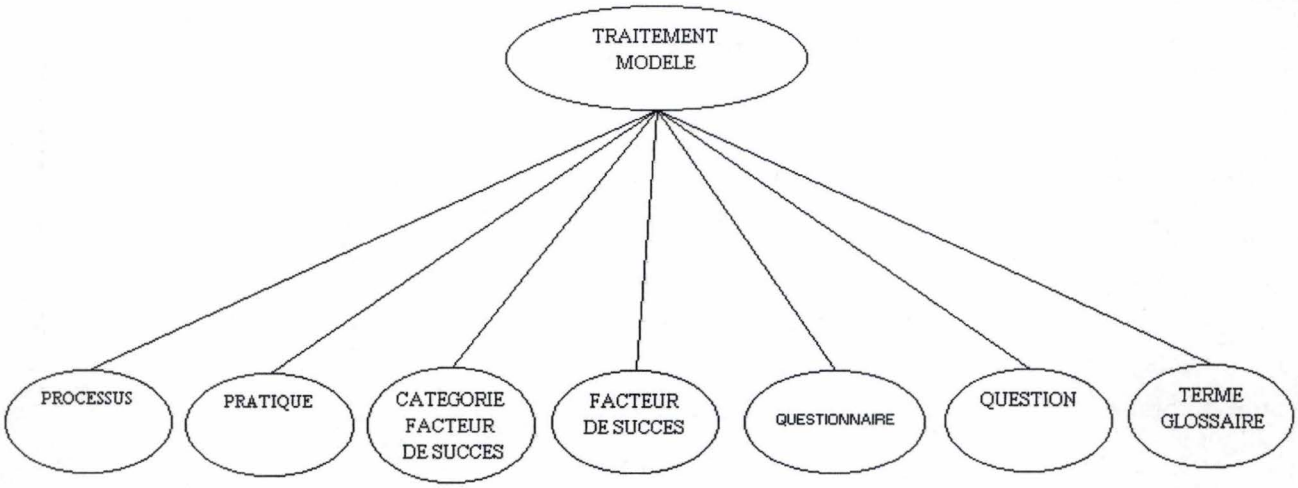
Figure 9 Schéma relationnel du modèle OWPL

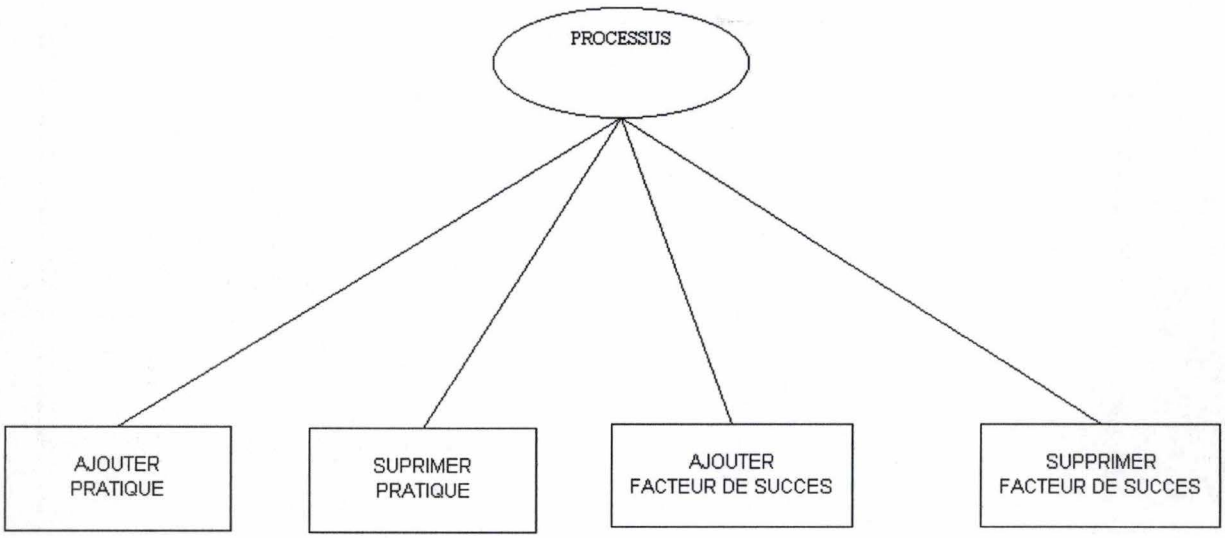
CHAPITRE 5. Architecture de l'application

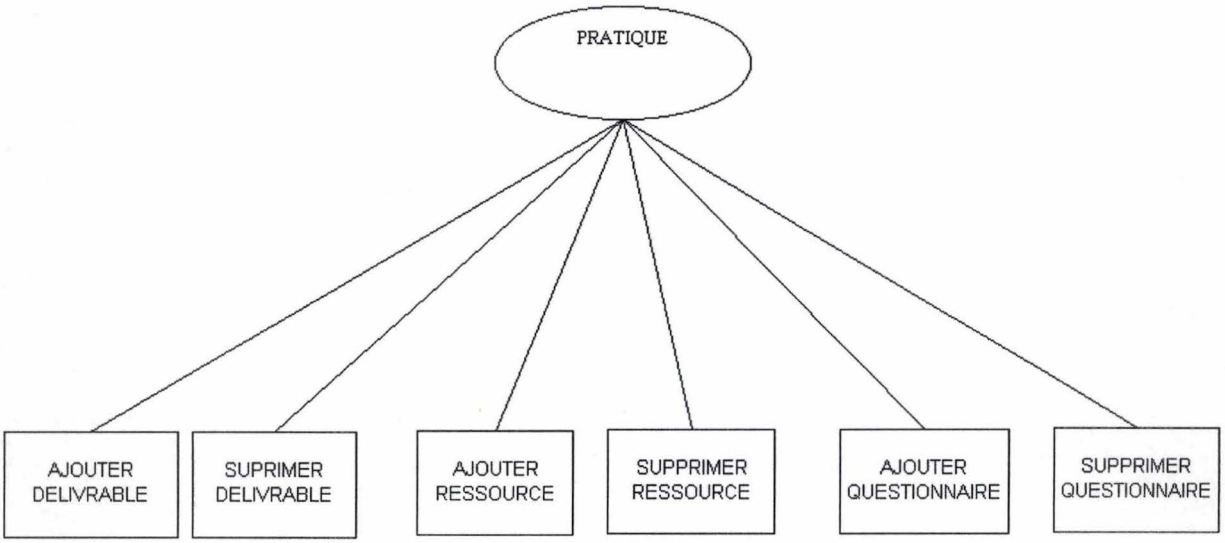
5.1. Découpe en sous-tâches de l'application

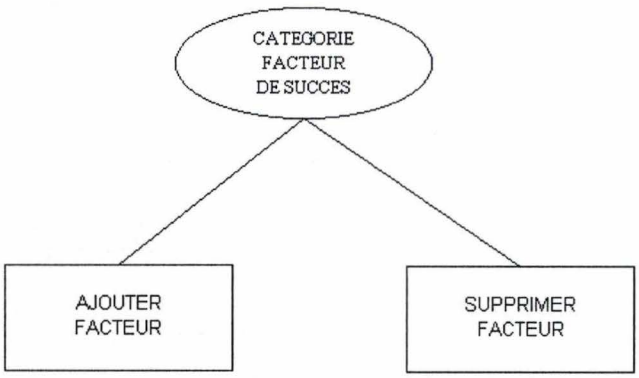
Dans cette partie, nous considérons d'abord l'application dans son entièreté, et ensuite, nous découpons cette dernière en différentes sous-tâches jusqu'à arriver à des fonctions.
A toutes les fonctions représentées ici, il faut ajouter pour toutes les entités les fonctions: ajout, suppression, modification d'un attribut et affichage de toutes les informations les concernant.
Les tâches ovales sont les tâches découpées en plusieurs sous-tâches tandis que les tâches rectangulaires sont des tâches qui ne le sont pas.

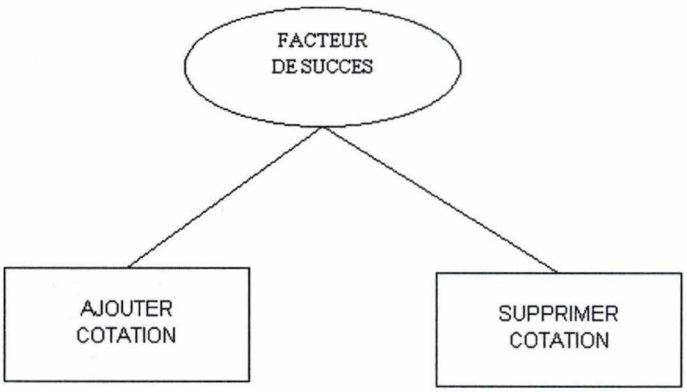


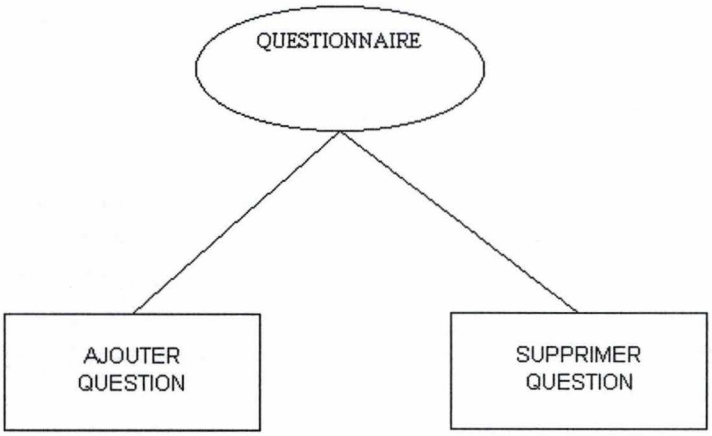

TRAITEMENT MODELE
PROCESSUS
PRATIQUE
CATEGORIE FACTEUR DE SUCCES
FACTEUR DE SUCCES
QUESTIONNAIRE
QUESTION
TERME GLOSSAIRE

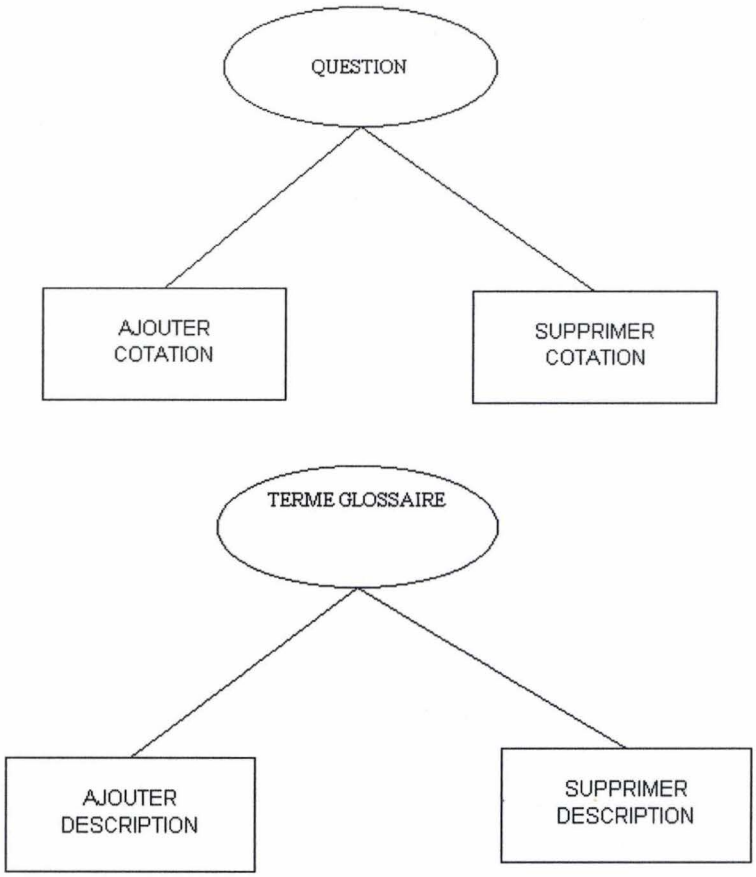

PROCESSUS
AJOUTER PRATIQUE
SUPRIMER PRATIQUE
AJOUTER FACTEUR DE SUCCES
SUPPRIMER FACTEUR DE SUCCES


PRATIQUE
AJOUTER DELIVRABLE
SUPRIMER DELIVRABLE
AJOUTER RESSOURCE
SUPPRIMER RESSOURCE
AJOUTER QUESTIONNAIRE
SUPPRIMER QUESTIONNAIRE


CATEGORIE
FACTEUR
DE SUCCES
AJOUTER
FACTEUR
SUPPRIMER
FACTEUR


FACTEUR
DE SUCCES
AJOUTER
COTATION
SUPPRIMER
COTATION


QUESTIONNAIRE
AJOUTER
QUESTION
SUPPRIMER
QUESTION



5.2. diagramme des flux

Le diagramme des flux permet de modéliser l'échange d'informations (des flux d'informations) entre différents acteurs qui participent a un processus (un système d'informations). Ce diagramme a l'avantage de donner les différents scénario d'utilisation du logiciel à implémenter.
Nous allons montrer ici de manière graphique, les entrées, les sorties des fonctions et l'interaction qui existe, éventuellement, entre elles.

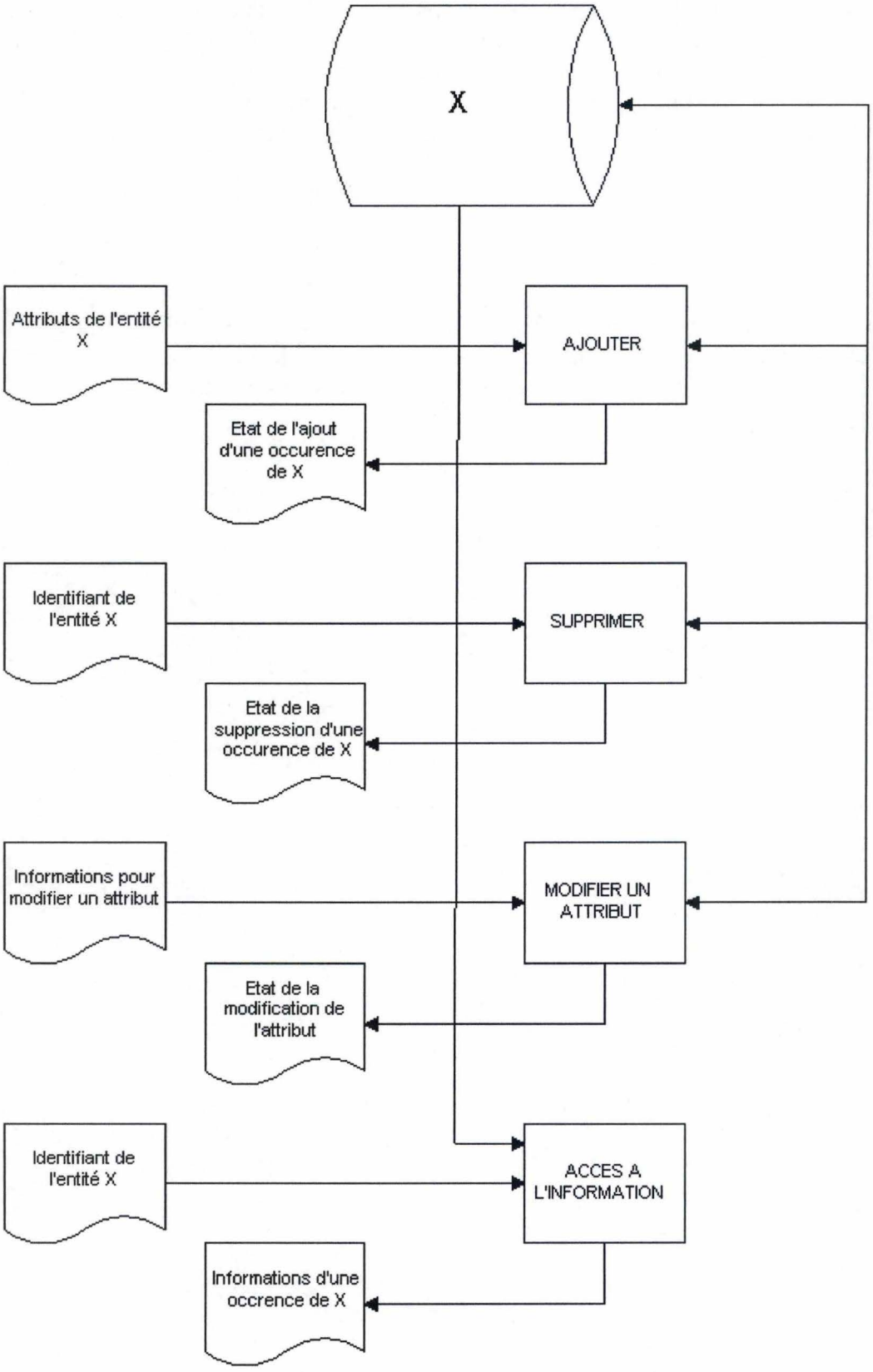
5.2.1 Conventions de représentations

Un trait plein partant d'une structure de données et allant vers une fonction signifie que la fonction puise des informations dans la structure en question. Si la flèche est dans l'autre sens, la fonction écrit ou met à jour des informations dans la structure. Un trait en pointillé signifie que la structure de données n'est pas toujours sollicitée.
Au ce niveau de développement, nous nous référons au schéma conceptuel défini au point 3.4. c'est la raison pour laquelle les attributs des différentes entités ne sont pas détaillées ici.

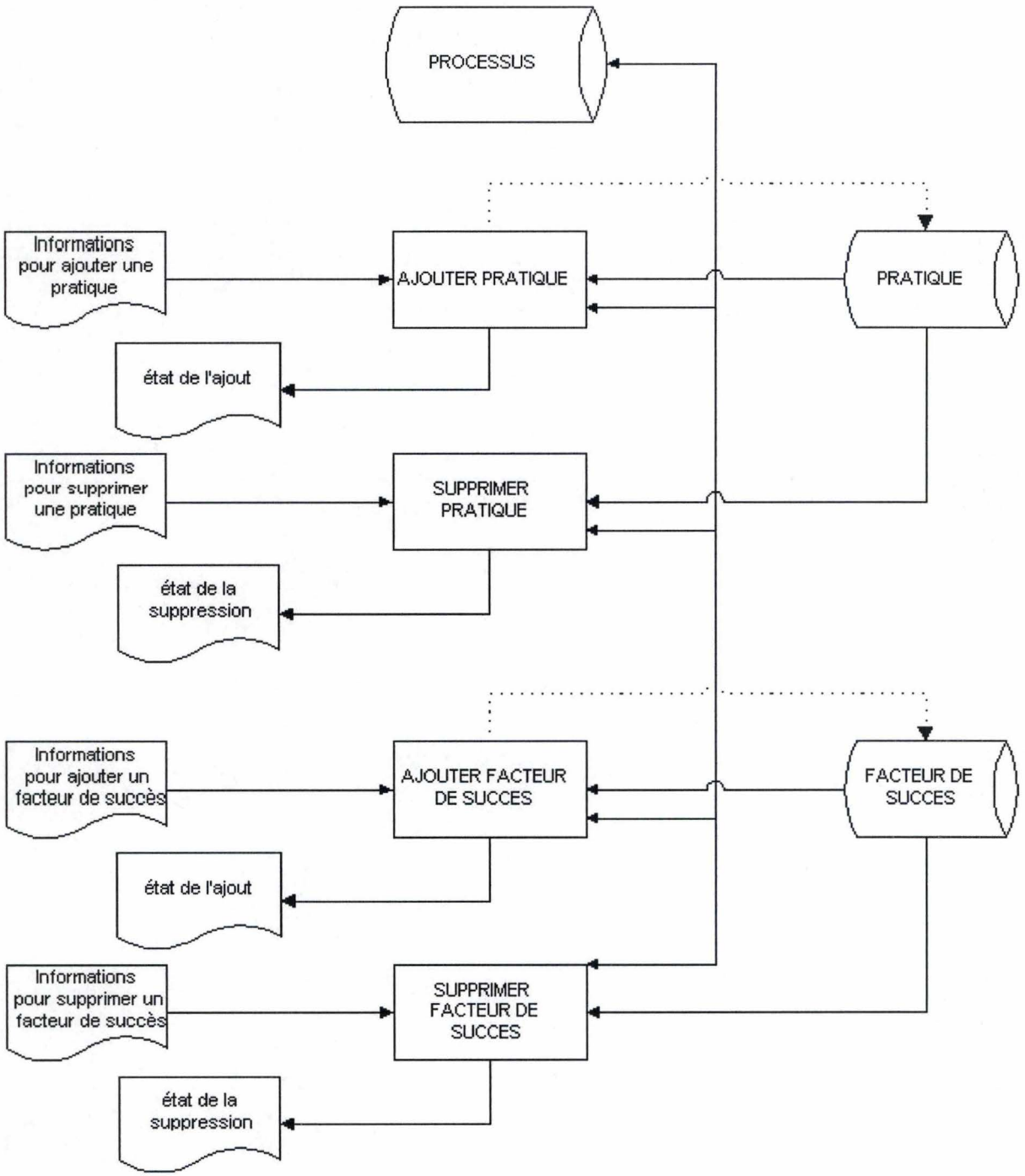
Exemple : pour une PRATIQUE, lorsque l'on veut ajouter une ressource, il se peut que cette ressource n'existe pas encore dans le SI. Dans ce cas, en plus de créer une association entre la pratique et la ressource, on crée la ressource. Le trait plein sur le schéma signifie que la fonction vérifie si l'identifiant de la ressource passé en paramètre existe dans le système. Le trait pointillé signifie que si un tel identifiant n'existe pas, alors une nouvelle occurrence d'une ressource est créé.

5.2.2 Fonctions génériques

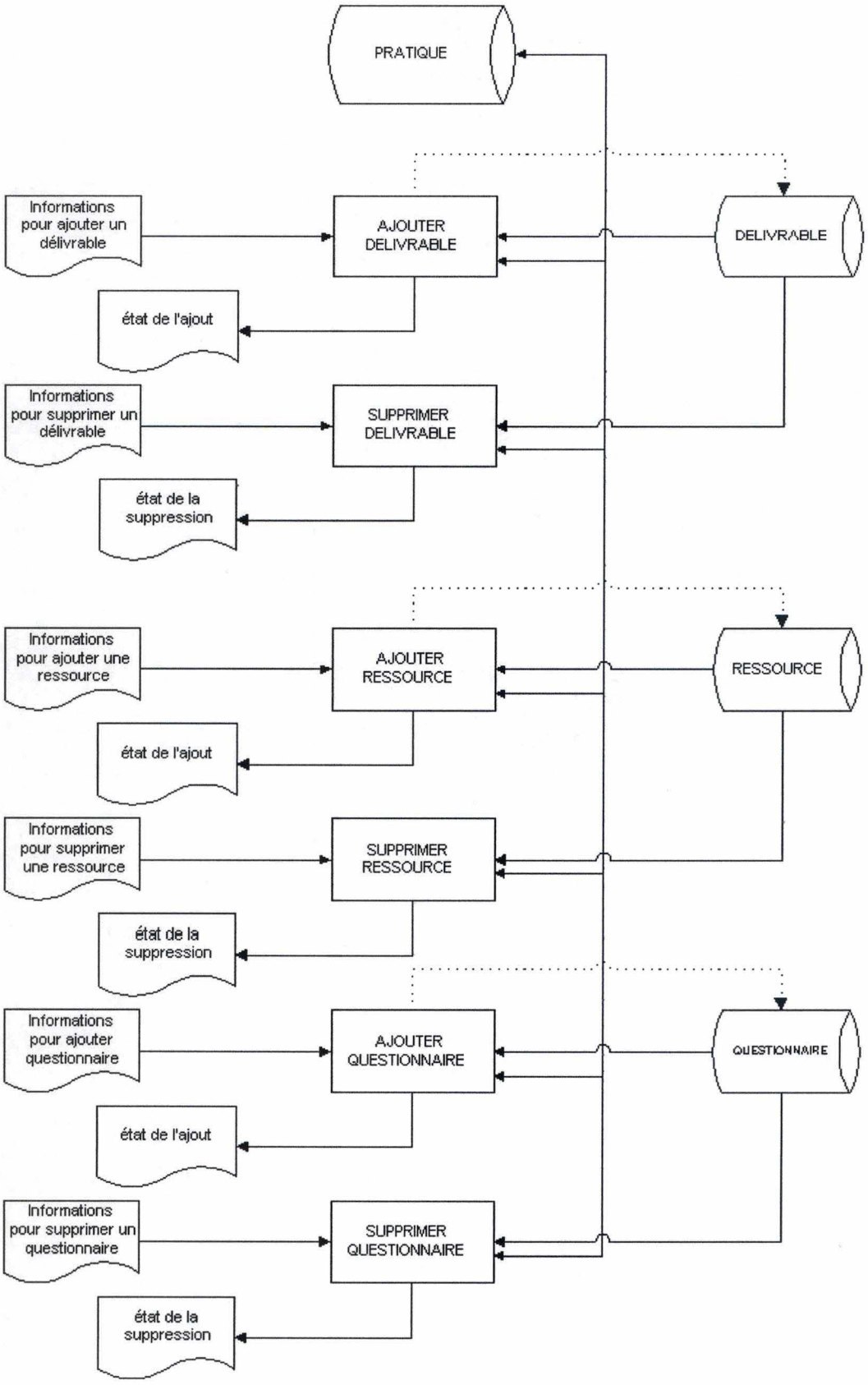
Toute entité du système possède au moins 4 fonctions de base : il s'agit de la création d'une nouvelle occurrence de l'entité, de la suppression, de la modification de l'un de ses attributs et enfin de l'accès à l'information. Ces 4 fonctions sont représentées pour une entité X générique quelconque.


X
Attributs de l'entité X
AJOUTER
Etat de l'ajout d'une occurence de X
Identifiant de l'entité X
SUPPRIMER
Etat de la suppression d'une occurence de X
Informations pour modifier un attribut
MODIFIER UN ATTRIBUT
Etat de la modification de l'attribut
Identifiant de l'entité X
ACCES A L'INFORMATION
Informations d'une occrence de X

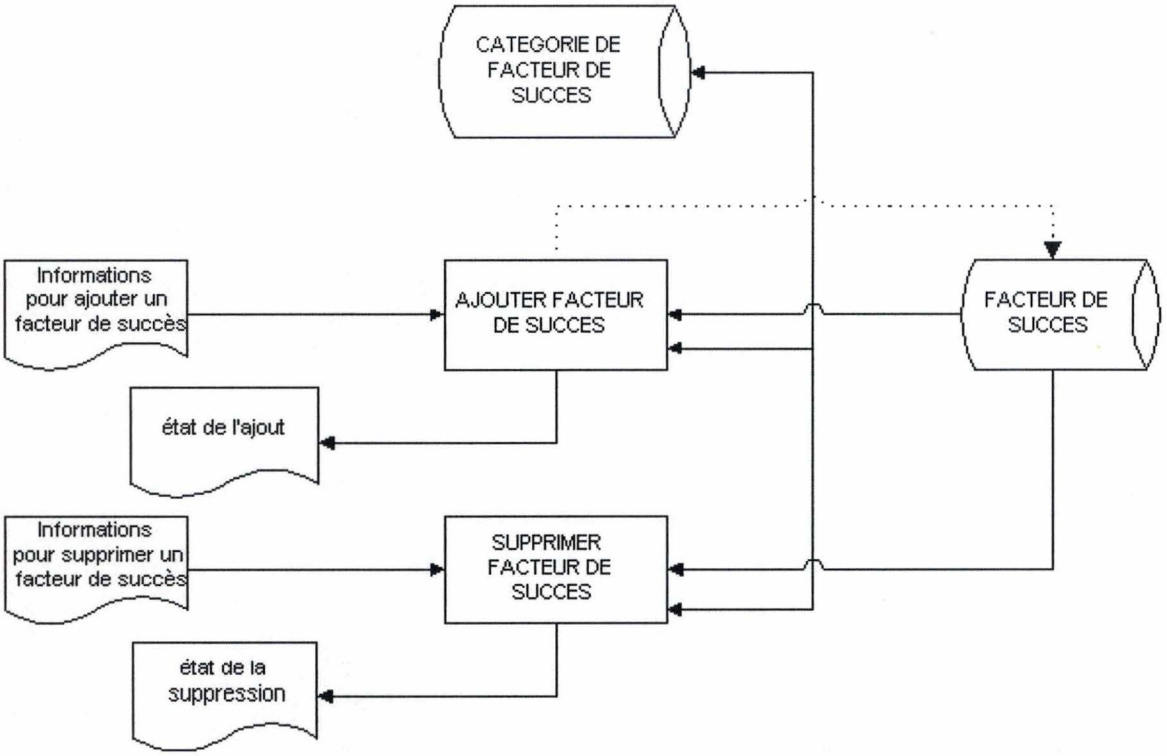
5.2.3 Processus



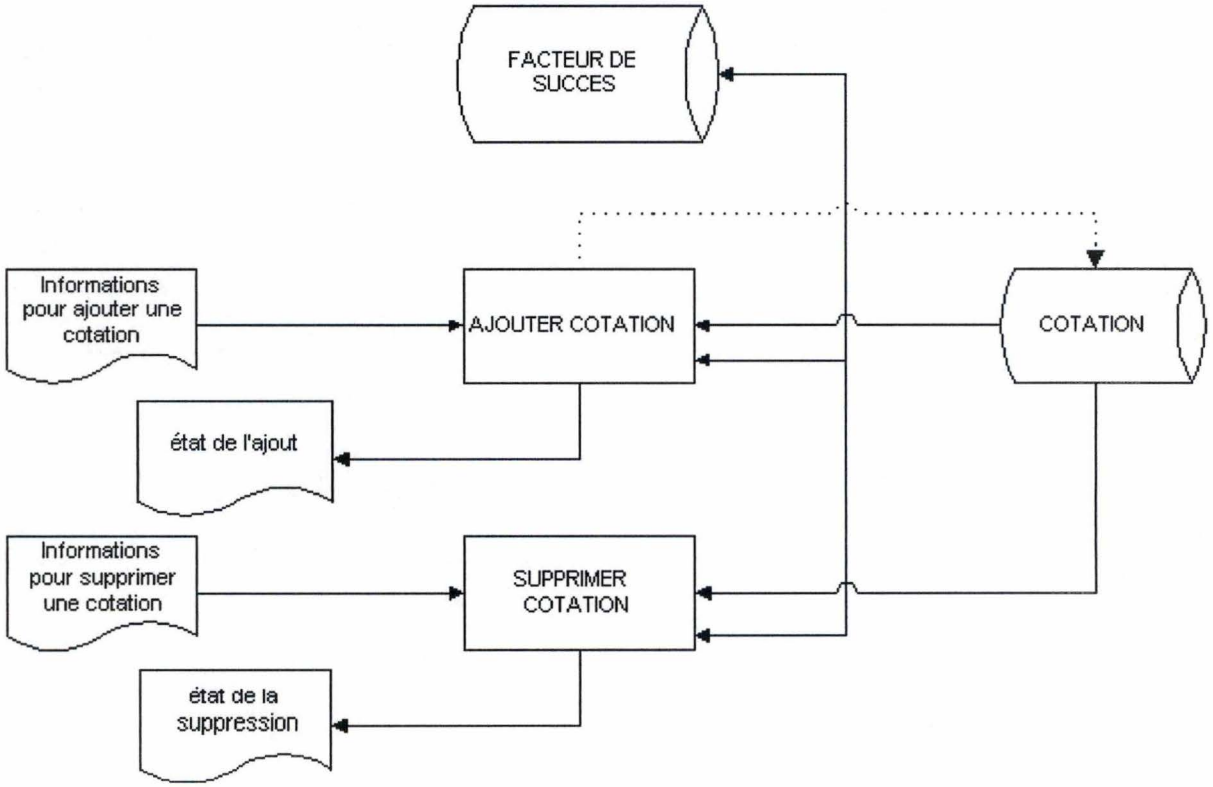
5.2.4 Pratique



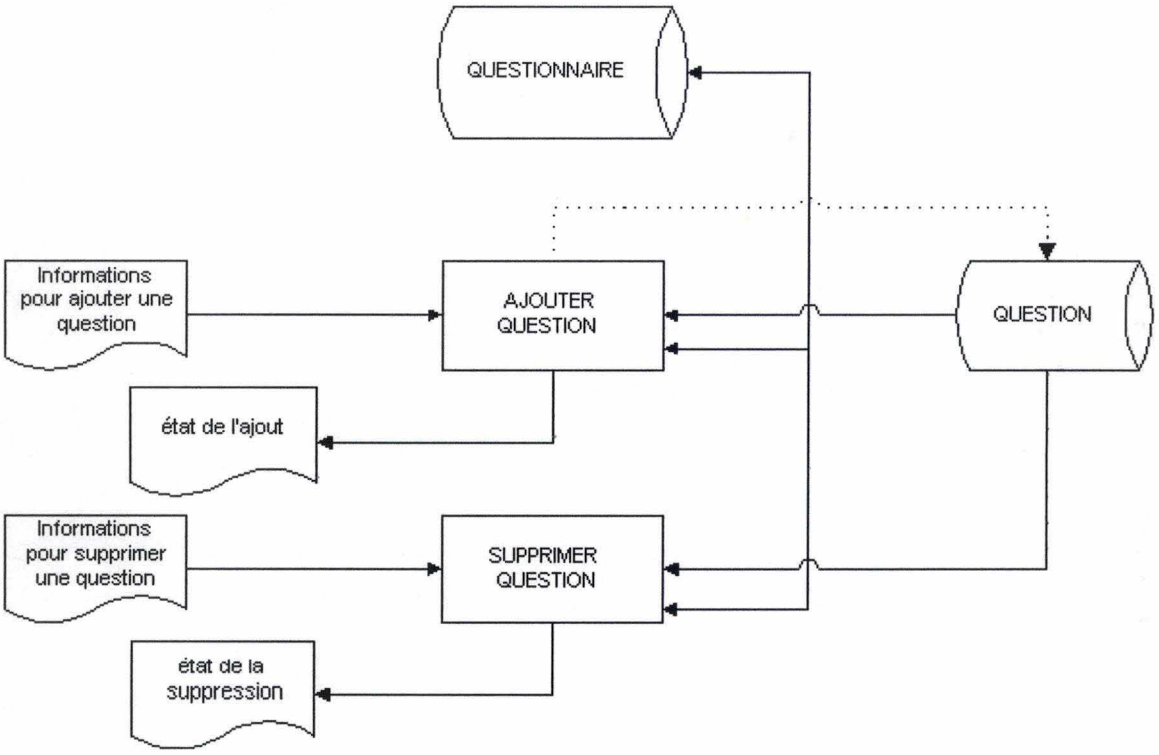
5.2.5 Catégorie facteur de succès



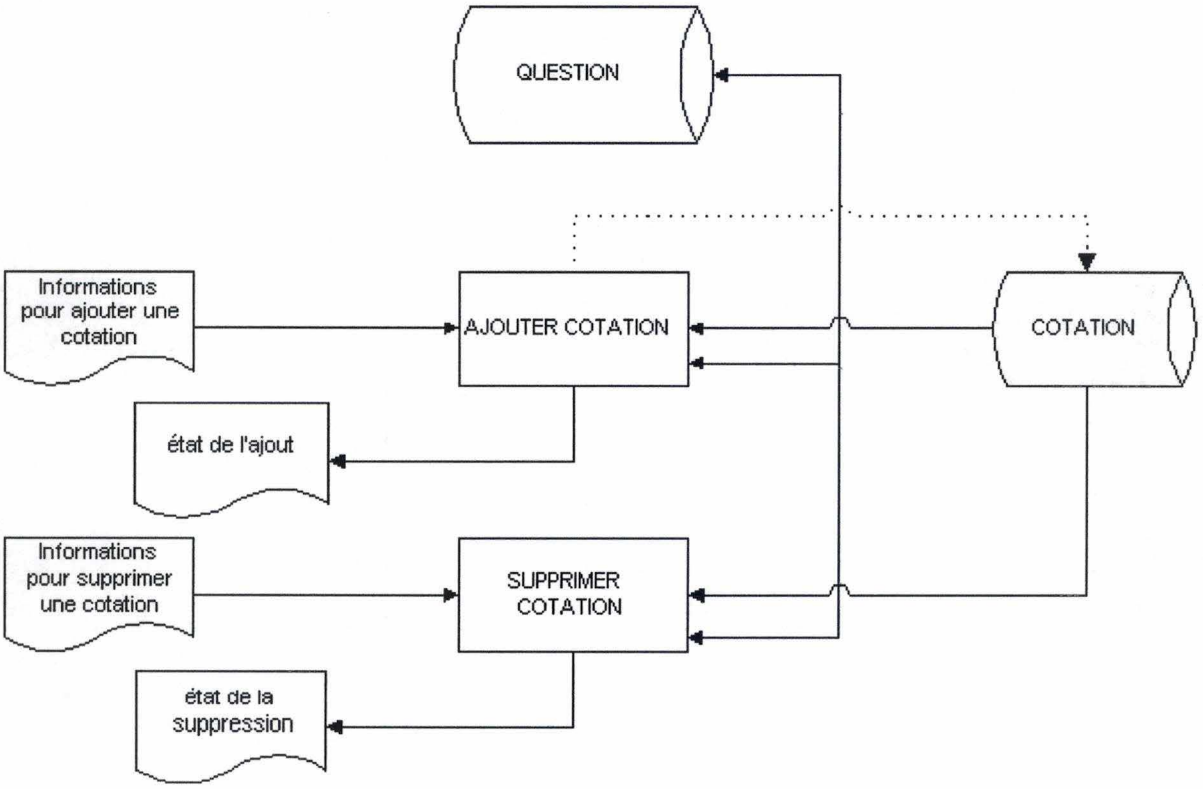
5.2.6 Facteur de succès



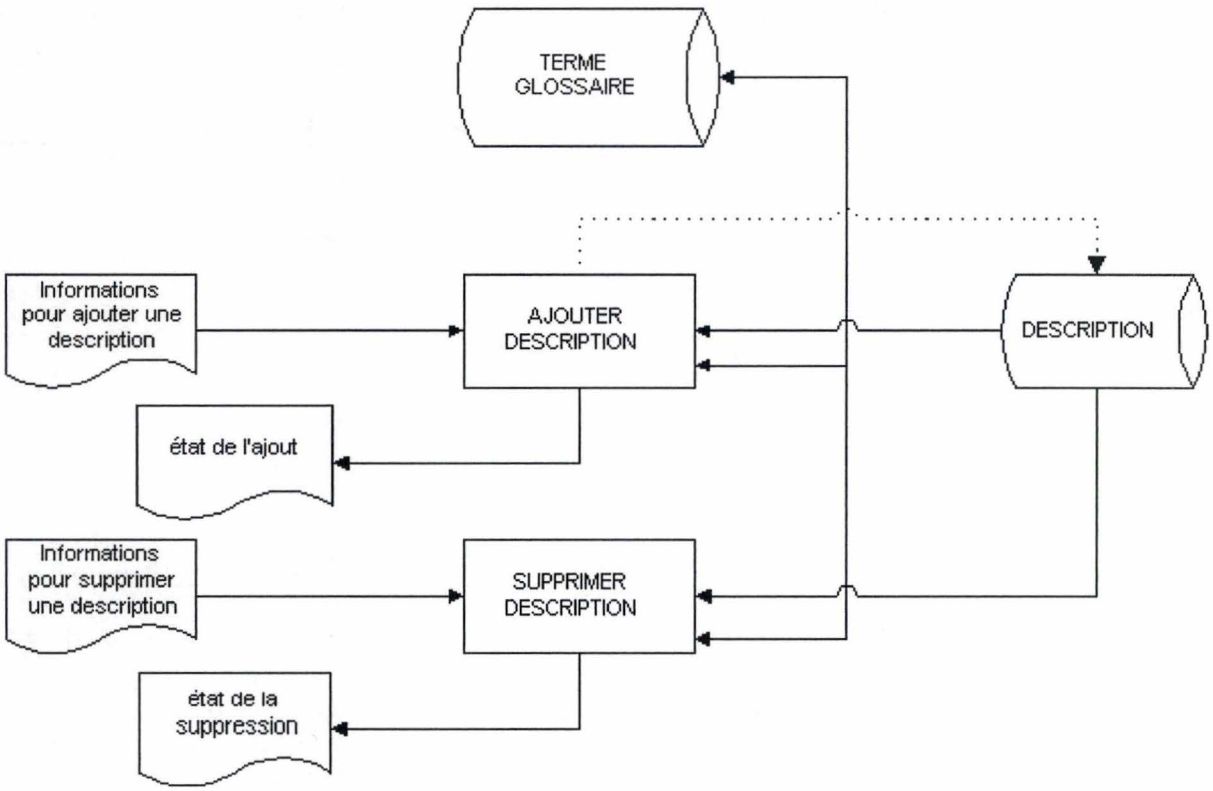
5.2.7 Questionnaire



5.2.8 Question



5.2.9 Terme glossaire



5.3. Spécification des fonctions

5.3.1 Conventions utilisées

La spécification des fonctions de l'application se fait ici suivant le schéma conceptuel de la base des données de l'application tel que défini au point 3.4. Ce schéma conceptuel a été fait sous forme d'un modèle Entité-Association[21]. C'est pourquoi les fonctions de l'application sont spécifiées de la manière suivante :

Nom de la fonction

Input : Attributs d'une entité quelconque Et/ou variables en entrée de la fonction Output : Attributs d'une entité quelconque Et/ou variables en sortie de la fonction Effet sur le SI : Il s'agit ici de l'effet de la fonction spécifiée sur le système d'information .

Il s'agit ici d'une fonction qui, suivant un état du système d'information, modifie la valeur d'un attribut soit d'une entité du système, soit d'une association du système. La fonction peut aussi créer une nouvelle association ou en supprimer une entre deux entités.
Dans le suite du présent document, toutes les entités citées sont les entités utilisées dans le Schéma Entité-Associations.

Il s'agit des entités : PROCESSUS, PRATIQUE, DELIVRABLE, RESSOURCE, CATEGORIE FACTEUR DE SUCCES, FACTEUR DE SUCCES, COTATION FACTEUR, QUESTIONNAIRE, QUESTION, PROPOSITION REPONSE, COTATION-QUESTION et TERME GLOSSAIRE.

Il s'agit des associations : Support, Appartenance, Composition, Entrées, Sorties, Utilisation, Concerne, Liste, Relative, Cotation-q, Cotation-f, et toutes les associations du genre Description-Entité.

5.3.2 Modèle OWPL

Etant donné la grande similitude de certaines fonctions, nous avons préféré faire de ces quelques fonctions des fonctions génériques. Celles-ci sont les fonctions d'ajout d'une entité, de suppression d'une entité, de mise à jour d'une entité pour un attribut non identifiant et les fonctions d'accès à toute information d'une entité, ou de n'importe quel attribut.

Il est évident aussi que toutes ces fonctions ont un input en plus. En effet, il faut ajouter le système d'information (Toute fonction prend en input la base des données). Toute modification du système d'information sera signalée sous le champ « effet sur le S.I. ».

[21] Voir le schéma conceptuel du modèle OWPL en annexe du présent document

Nous avons décidé de ne pas spécifier ici la fonction de gestion d'erreur nécessaire pour toute saisie d'information. On fait chaque fois l'hypothèse ici que les données sont syntaxiquement valides. La gestion des erreurs sera spécifiée suivant l'outil de développement qui sera choisi. Cette gestion des erreurs consiste en une fonction qui prend en input des données concernant une entité et a comme output, une variable booléenne qui atteste de la validité de ces données suivant la définition de l'entité concernée. C'est donc une pré-condition générale à toutes les fonctions.

Ajout d'une entité X

Input :
des valeurs syntaxiquement valides pour tous les attributs obligatoires de l'entité X, et éventuellement pour les attributs facultatifs.
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une nouvelle occurrence de l'entité X.

Si on ajoute une entité et qu'elle fait l'objet d'une d'association dont le rôle qui lui correspond est obligatoire, il est incontestable qu'il faut aussi créer une occurrence de cette association. Cela ne sera pas décrit non plus dans les spécifications fonctionnelles.

Suppression d'une entité X

Input :
ID_X : identifiant de l'entité X que l'on désire supprimer.
Output :
-

Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ diminuées de l'occurrence de l'entité X dont l'identifiant était ID_X, l'identifiant apporté en input.

En ce qui concerne la suppression d'une entité particulière qui aurait des interactions avec d'autres, il nous a paru opportun de décider d'un mode de suppression en cascade, c'est-à-dire que la suppression d'une telle entité entraînera la suppression de toutes les entités et associations en rapport avec elle qui doivent subir une suppression en cascade. Il est évident que ne seront pas supprimés les entités encore nécessaires(Par exemple, lors de la suppression d'une pratique, les délivrables encore utilisés par d'autres pratiques ne sont pas supprimés). La gestion de ce mode de suppression ne sera pas spécifié dans les descriptions des fonctions.

Mise à jour de l'attribut Y de l'entité X

Input :
ID_X : identifiant de l'entité X
ATT_Y : un attribut Y de l'entité X

VAL_Y : une valeur syntaxiquement valide pour l'attribut Y de l'entité X

Output :

-

Effet sur le S.I. :

le nouvel état du système d'information contient les mêmes informations que l'état de départ sauf pour l'occurrence de l'entité X identifiée par ID_X pour laquelle l'attribut ATT_Y est mis à VAL_Y.

Accès aux valeurs des ATT_Y de l'entité X

Input :

X : nom d'une entité

ATT_Y : un attribut Y de l'entité X dont on désire connaître toutes les valeurs

Output :

liste de toutes les valeurs de ATT_Y de l'entité X

Effet sur le S.I. :

-

5.3.3 Processus

Ajout d'une pratique à un processus

Input :

ID_PT : identifiant d'une pratique

ID_PC : identifiant d'un processus

Output :

-

Effet sur le S.I. :

le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association COMPOSITION liant le PROCESSUS identifié par ID_PC à la PRATIQUE identifiée par ID_PR.

Suppression d'une pratique d'un processus

Input :

ID_PT : identifiant d'une pratique

ID_PC : identifiant d'un processus

Output :

-

Effet sur le S.I. :

le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association COMPOSITION liant le PROCESSUS identifié par ID_PC à la PRATIQUE identifiée par ID_PR qui a été supprimée.

L'association COMPOSITION étant obligatoire pour une pratique, supprimer une pratique d'un processus revient en même temps à supprimer la pratique du modèle, étant donné qu'elle n'est plus liée à aucun processus.

Ajout d'un facteur de succès à un processus

Input :
ID_F : identifiant d'un facteur de succès
ID_PC : identifiant d'un processus
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association SUPPORT liant le FACTEUR DE SUCCES identifié par ID_F au PROCESSUS identifié par ID_PC.

On considère ici aussi bien lors de l'ajout que e la suppression, un support particulier qu'un facteur de succès peut constituer pour une pratique. Ce support est particulier dans la mesure où, dans le modèle OWPL, tous les facteurs de succès supportent en général toutes les pratiques.

Suppression d'un facteur de succès d'un processus

Input :
ID_F : identifiant d'un facteur de succès
ID_PC : identifiant d'un processus
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association SUPPORT liant le FACTEUR DE SUCCES identifié par ID_F au PROCESSUS identifié par ID_PC qui est supprimée.

5.3.4 Pratique

Ajout d'un délivrable à une pratique

Input :
ID_PT : identifiant d'une pratique
ID_D : identifiant d'un délivrable
Ass ∈ {Entrée, Sortie}
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association ASS liant le DELIVRABLE identifié par ID_D à la PRATIQUE identifiée par ID_PR.

Suppression d'un délivrable d'une pratique

Input :
ID_PT : identifiant d'une pratique
ID_D : identifiant d'un délivrable
Ass ∈ {Entrée, Sortie}
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association Ass liant le DELIVRABLE identifié par ID_D à la PRATIQUE identifiée par ID_PR qui est supprimée.

Ajout d'une ressource à une pratique

Input :
ID_PT : identifiant d'une pratique
ID_R : identifiant d'une ressource
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association UTILISATION liant la RESSOURCE identifiée par ID_R à la PRATIQUE identifiée par ID_PR.

Suppression d'une ressource d'une pratique

Input :
ID_PT : identifiant d'une pratique
ID_R : identifiant d'une ressource
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association UTILISATION liant la RESSOURCE identifiée par ID_R à la PRATIQUE identifiée par ID_PR qui est supprimée.

Ajout d'un questionnaire à une pratique

Input :
ID_Q : identifiant d'un questionnaire
ID_PR : identifiant d'une pratique
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association CONCERNE liant le QUESTIONNAIRE identifié par ID_Q à la PRATIQUE identifiée par ID_PR.

Suppression d'un questionnaire à une pratique

Input :
ID_Q : identifiant d'un questionnaire
ID_PR : identifiant d'une pratique
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association CONCERNE liant le QUESTIONNAIRE identifié par ID_Q à la PRATIQUE identifiée par ID_PR qui est supprimée.

5.3.5 Catégorie facteur de succès

Ajout d'un facteur de succès à une catégorie de facteur de succès

Input :
ID_C : identifiant d'une catégorie de facteur de succès
ID_F : identifiant d'un facteur de succès
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association APPARTENACE liant la CATEGORIE DE FACTEUR DE SUCCES identifiée par ID_C au FACTEUR DE SUCCÈS identifié par ID_F.

Suppression d'un facteur de succès d'une catégorie de facteur de succès

Input :
ID_C : identifiant d'une catégorie de facteur de succès
ID_R : identifiant d'un facteur de succès
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association APPARTENANCE liant la CATEGORIE DE FACTEUR DE SUCCES identifiée par ID_C au FACTEUR DE SUCCES identifié par ID_F qui est supprimée.

Comme pour les pratiques, l'association APPARENANCE étant obligatoire pour une facteur de succès, supprimer un facteur de succès d'une catégorie de facteur de succès revient à supprimer le facteur de succès du modèle.

5.3.6 Facteur de succès

Ajout d'une cotation à un facteur de succès

Input :
ID_F : identifiant d'un facteur de succès
ID_C : identifiant d'une cotation
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association COTATION-F liant le FACTEUR DE SUCCES identifié par ID_F à la COTATION identifiée par ID_C.

Suppression d'une cotation d'un facteur de succès

Input :
ID_F : identifiant d'un facteur de succès
ID_C : identifiant d'une cotation
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association COTATION-F liant le FACTEUR DE SUCCES identifiée par ID_F à la COTATION identifiée par ID_C qui est supprimée.

5.3.7 Questionnaire

Ajout d'une question à un questionnaire

Input :
ID_Q : identifiant d'une question
ID_QN : identifiant d'un questionnaire
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association LISTE liant la QUESTION identifiée par ID_Q au QUESTIONNAIRE identifié par ID_QN.

Suppression d'une question à un questionnaire

Input :

ID_Q : identifiant d'une question
ID_QN : identifiant d'un questionnaire

Output :

-

Effet sur le S.I. :

le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association LISTE liant la QUESTION identifiée par ID_Q au QUESTIONNAIRE identifié par ID_QN qui est supprimée.

La remarque sur les pratiques et les facteurs de succès est également valable pour les question, vu le caractère obligatoire de leur association LISTE.

5.3.8 Question

Ajout d'une cotation à une question

Input :

ID_Q : identifiant d'une question
ID_C : identifiant d'une cotation

Output :

-

Effet sur le S.I. :

le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association COTATION-Q liant la QUESTION identifiée par ID_Q à la COTATION identifiée par ID_C.

Suppression d'une cotation d'une question

Input :

ID_Q : identifiant d'une question
ID_C : identifiant d'une cotation

Output :

-

Effet sur le S.I. :

le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association COTATION-Q liant la QUESTION identifiée par ID_Q à la COTATION identifiée par ID_C qui est supprimée.

Ajout d'une proposition de réponse à une question

Input :

ID_Q : identifiant d'une question
ID_P : identifiant d'une proposition-réponse

Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association RELATIVE liant la QUESTION identifiée par ID_Q à la PROPOSITION-REPONSE identifiée par ID_P.

Suppression d'une proposition de réponse d'une question

Input :
ID_Q : identifiant d'une question
ID_P : identifiant d'une proposition-réponse
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association RELATIVE liant la QUESTION identifiée par ID_Q à la PROPOSITION-REPONSE identifiée par ID_P qui est supprimée.

5.3.9 Terme-glossaire

Pour les entités PROCESSUS, FACTEUR DE SUCCES, CATEGORIE DE FACTEUR DE SUCCES, PRATIQUE, DELIVRABLE, et RESSOURCE, nous définissons une fonction générique qui établit ou supprime une association de type Description-ENTITE avec l'entité TERME GLOSSAIRE.

Ajout d'une description du glossaire

Input :
ID_T : identifiant d'un terme du glossaire
ID_X : identifiant de l'entité X qu'on veut décrire.
X : entité qu'on utilise
Output :
-
Effet sur le S.I. :
le nouvel état du système d'information contient les mêmes informations que l'état de départ augmenté d'une association DESCRIPTION-X liant l'entité X identifiée par ID_X au TERME GLOSSAIRE identifié par ID_T et L'attribut facultatif Description de l'entité X identifiée par ID_X prend la valeur NULL si elle en avait une.

Suppression d'une description du glossaire

Input :
ID_T : identifiant d'un terme du glossaire
ID_X : identifiant d'une entité X qui est décrit dans le glossaire

X : entité qu'on utilise

Output :

-

Effet sur le S.I. :

le nouvel état du système d'information contient les mêmes informations que l'état de départ exceptée l'association DESCRIPTION-X liant l'entité X identifiée par ID_X au TERME DU GLOSSAIRE identifié par ID_T qui est supprimée.

CHAPITRE 6. L'application OWPL Manager

6.1.1 Fonctionnalités développées

Comme nous l'avons précisé dans l'introduction, seule une partie de l'application a été développée, en raison du temps qui nous était imparti. En partant de la découpe en sous-tâches, seule la partie concernant la gestion du modèle OWPL et du glossaire a été implémentée.

Avec cette partie, on peut gérer tout le modèle OWPL et le glossaire. Cette implémentation partielle s'est faite sans complication, car il y a une indépendance entre les différentes sous-tâches de l'application.

D'après les exigences de l'utilisateur, on peut dire que les fonctions suivantes ont été développées :

Pour les processus:

- Création d'un nouveau processus
- Consultation d'un processus
- Suppression d'un processus
- Modification d'un processus

Pour les pratiques:

- Création d'une pratique
- Consultation d'une pratique
- Suppression d'une pratique
- Modification d'une pratique
- ajout d'un délivrable à une pratique
- suppression d'un délivrable d'une pratique
- ajout d'une ressource à une pratique
- suppression d'une ressource d'une pratique

Pour les délivrables:

- création d'un nouveau délivrable
- suppression d'un délivrable du modèle

Pour les ressources:

- création d'une nouvelle ressource
- suppression d'une ressource du modèle

Pour les termes du glossaire:

- création d'un nouveau terme
- consultation d'un terme
- modification de la définition d'un terme
- suppression d'un terme

6.1.2 Techniques utilisées

Le parseur ProjectX de chez Sun est utilisé au lancement de l'application pour "charger" la base des données. Ce "chargement" de la base des données se fait via le parsing des différents fichiers XML avec l'interface DOM du parseur. Avec les différents arbres obtenus, le contenu de chaque arbre est recopié dans un vecteur Java. Le contenu de chaque fichier XML est transféré dans un vecteur qui contient exactement les mêmes données, et garde la structure du fichier XML. Les vecteurs sont utilisés à la place des tableaux classiques grâce à leur aspect dynamique.

Ces vecteurs Java sont utilisés par l'application comme la base des données chargée en mémoire, et c'est seulement quand l'utilisateur veut sauvegarder ses dernières transformations que l'application écrase les fichiers XML. Il est aussi possible à l'utilisateur d'exporter la dernière version modifiée du modèle sous la forme des pages HTML, ceci pour permettre une consultation du modèle OWPL et du glossaire en mode lecture simple.

A part quelques classes Java qui implémentent les fonctions génériques utilisées par toutes les classes, à chaque table de la base des données (entendez le fichier XML correspondant) correspond une classe Java. L'API des classes utilisées est donné en annexe (document n°6) sous-forme de fichiers HTML générés avec l'outil javadoc.

6.1.3 L'application OWPL Manager

L'application développée a été baptisée *OWPL Manager*. Au niveau actuel de développement décrit plus haut, on peut dire qu'on est à sa version 1.0.0. Nous allons voir cette application dans ses deux parties essentielles : la partie gestion du modèle OWPL qui se fait avec une application Java, et la partie consultation du modèle OWPL qui se fait grâce à une page web.

6.1.3.1 Gestion du modèle OWPL

Pour cette version, l'utilisateur est accueilli avec la fenêtre présentée sur la figure 10.

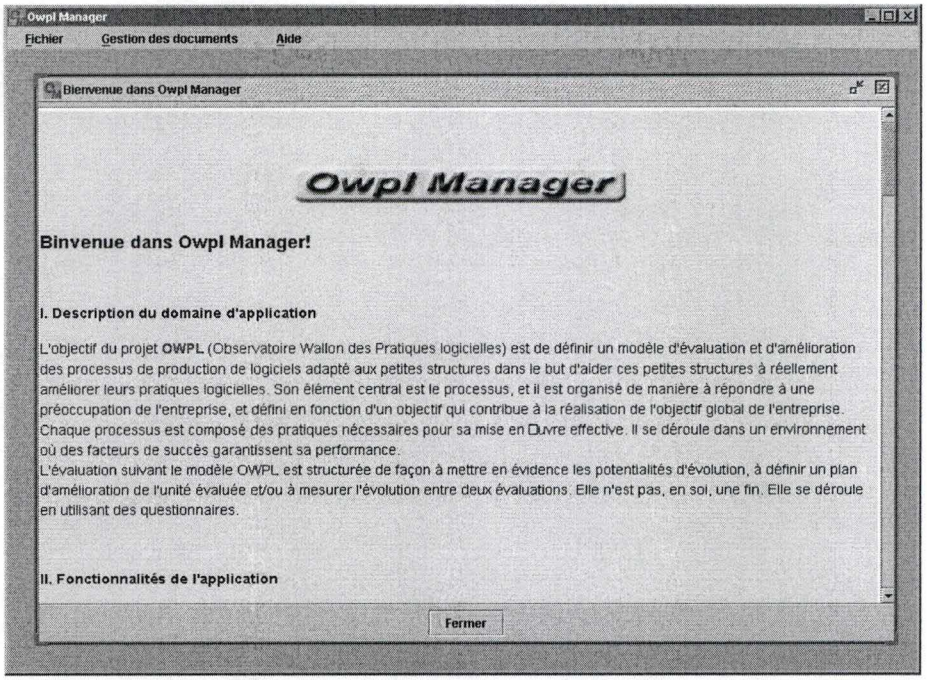


Figure 10: Fenêtre de départ de l'application OWPL Manager

Dans cette fenêtre, une petite fenêtre d'accueil interne présente le projet OWPL. Après lecture, l'utilisateur peut la fermer en appuyant sur le bouton "Fermer".

Trois menus déroulant sont disponibles pour l'utilisateur :

Le menu "Fichier" (figure 11) permet d'accéder au modèle OWPL, au glossaire, aux facteurs de succès, aux questionnaires et aussi. Il permet aussi de sauver l'état de la base des données en mémoire, d'exporter cette base des données en fichiers HTML, ou de quitter simplement l'application.
Au stade actuel de l'implémentation, l'utilisateur n'a accès qu'au modèle OWPL, au glossaire, à la sauvegarde des fichiers XML et à l'exportation de la base des données sous forme de pages HTML.

Le menu "Gestion des documents" (figure 12) permet de générer des documents écrits sur le modèle OWPL. Cette fonctionnalité n'est pas encore implémentée.

Le menu "Aide" (figure 13) permet d'obtenir les information sur le projet OWPL et sur l'application OWPL Manager. Les informations sur le projet OWPL sont données par la fenêtre d'accueil interne d'OWPL Manager. C'est donc cette fenêtre qui est affichée quand il appuie sur "Le modèle OWPL".

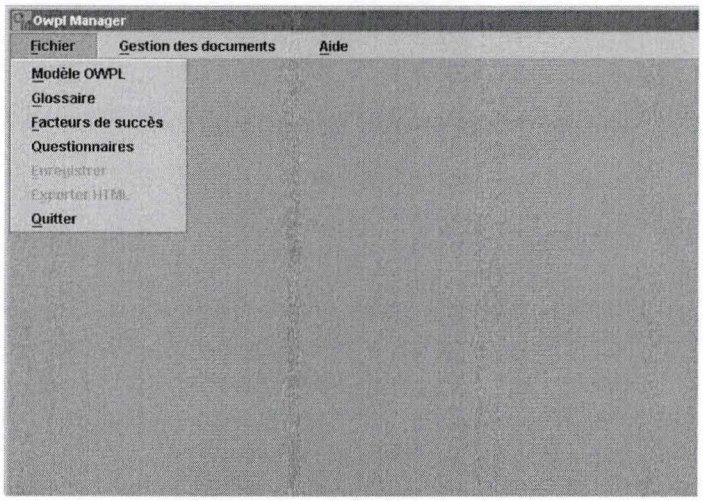


Figure 11: Menu Fichier de l'application OWPL Manager

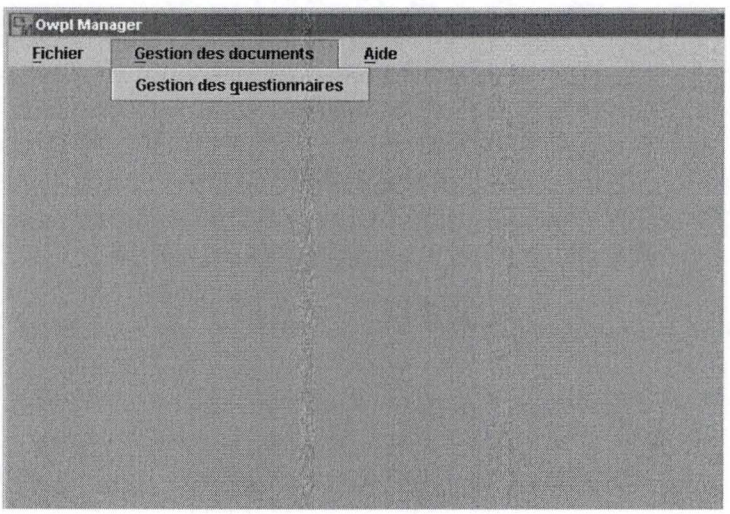


Figure 12: Menu Gestion des documents de l'application OWPL Manager

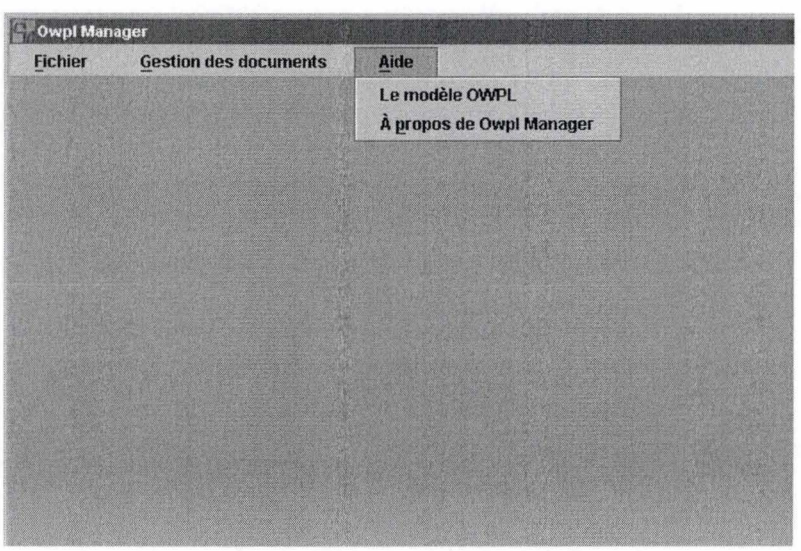


Figure 13: Menu Aide de l'application OWPL Manager

L'utilisateur qui appuie sur "Le Modèle OWPL" du menu fichier arrive à la fenêtre de la figure 14 :

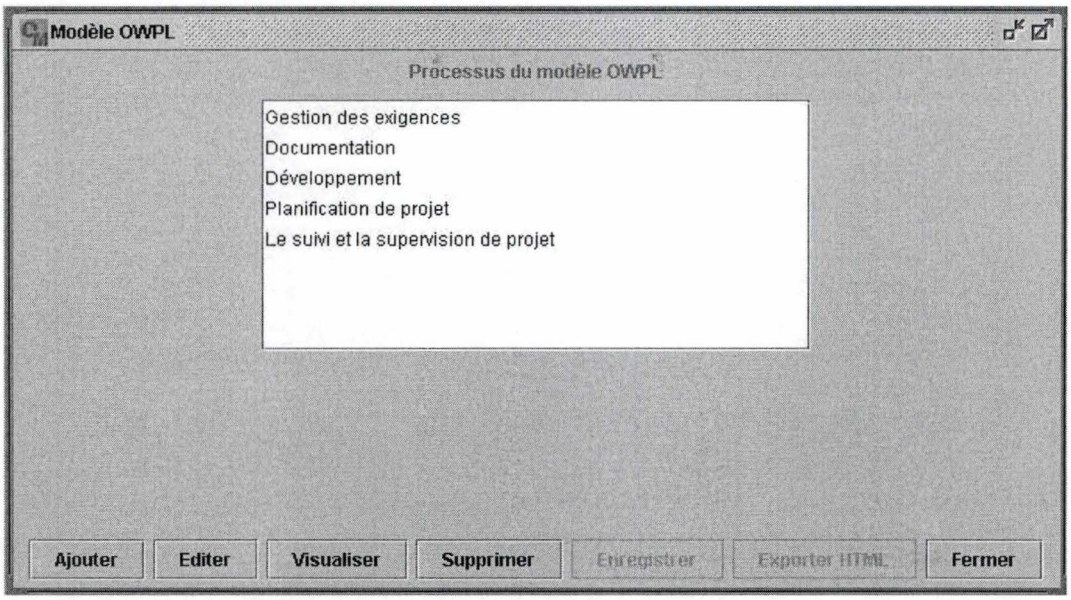


Figure 14: Les processus du modèle OWPL dans OWPL Manager

Il s'agit de la présentation de tous les processus du modèle qu'on peut alors éditer, visualiser ou supprimer. Il y a aussi possibilité d'ajout d'un nouveau processus.

Aussi bien dans le cas d'un ajout, d'une édition que d'une visualisation d'un processus, l'utilisateur dans une fenêtre à celle de la figure 15. Sur la figure 15, il s'agit de la fenêtre d'édition d'un processus.
A partir de la fenêtre d'édition d'un processus, l'utilisateur peut modifier les données relatives au processus et manipuler les pratiques du processus édité. La manipulation des pratiques passe par leur ajout, édition, visualisation ou leur suppression.

Comme pour un processus, l'ajout, l'édition ou la visualisation d'une pratique conduit à une fenêtre identique à celle de la figure 16. La figure 16 représente l'édition d'une pratique du modèle OWPL.

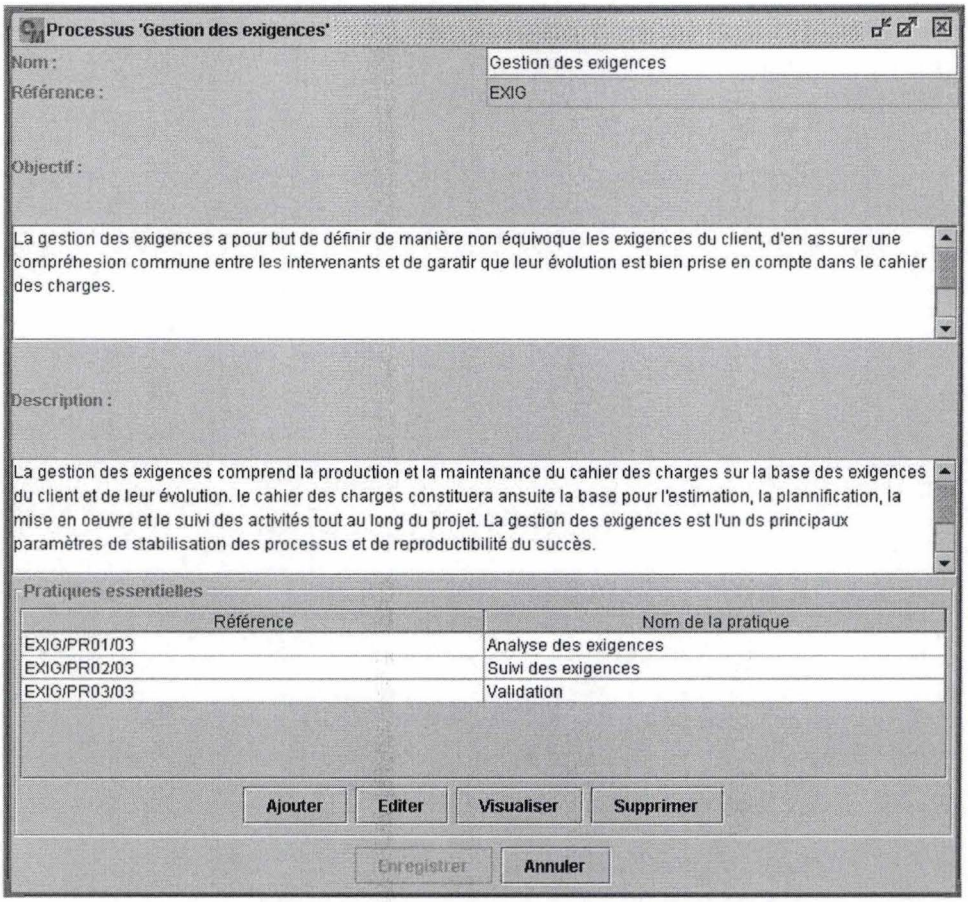


Figure 15: Edition d'un processus dans OWPL Manager

Lors de l'édition d'une pratique comme illustrée par la figure 16, on peut modifier toutes les données relatives à la pratique, ajouter ou supprimer une entrée, une sortie ou une ressource de la pratique.

En cas d'ajout d'un délivrable, l'utilisateur se trouve face à une liste de tous les délivrables du modèle OWPL triés par ordre croissant sur leurs noms. Il est devant la fenêtre illustré par la figure 17, où, s'il ne trouve pas le délivrable voulu, il peut en créer un nouveau. S'il trouve le délivrable voulu, il peut l'ajouter à la pratique à partir de laquelle il a appelée la fenêtre active.

Il a également la possibilité d'éditer, de visualiser ou de supprimer un délivrable. Un scénario identique se produit quand l'utilisateur veut ajouter ou supprimer une ressource d'une pratique.

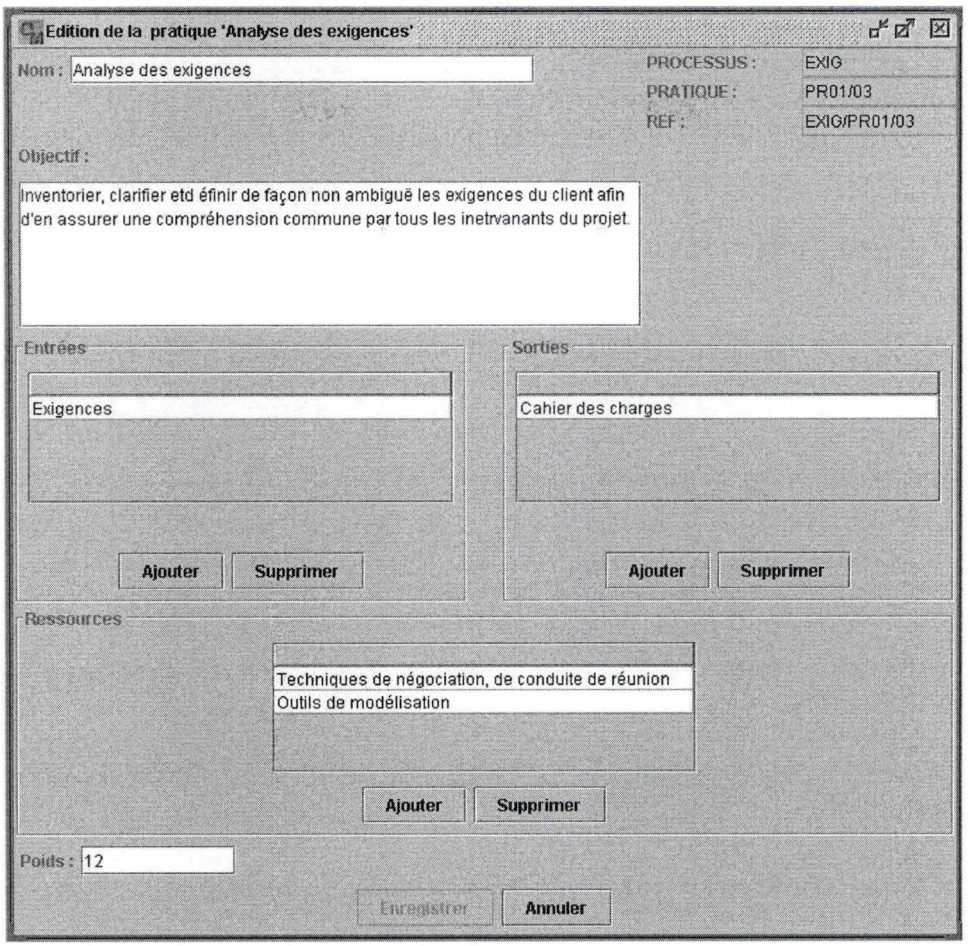


Figure 16: Edition d'une pratique dans OWPL Manager

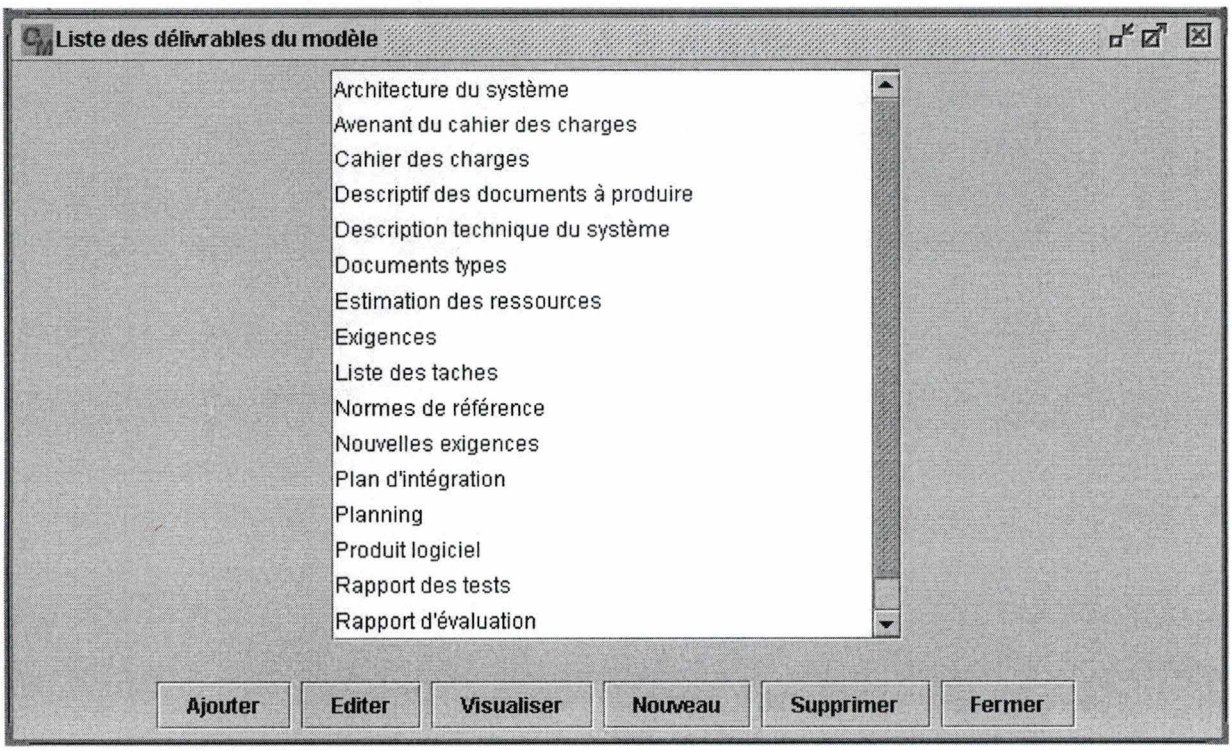


Figure 17: Liste des délivrables du modèle OWPL dans OWPL Manager

Pour ce qui concerne le glossaire, l'utilisateur qui appuie sur "Glossaire" (figure 11) dans le menu fichier se trouve devant la fenêtre de le figure 18. Il s'agit d'une fenêtre qui présente l'état du glossaire, tout en permettant d'ajouter un terme, de l'éditer ou de le supprimer.

La fenêtre d'édition d'un terme est identique çà la fenêtre d'ajout d'un terme et est représentée par la figure 19.

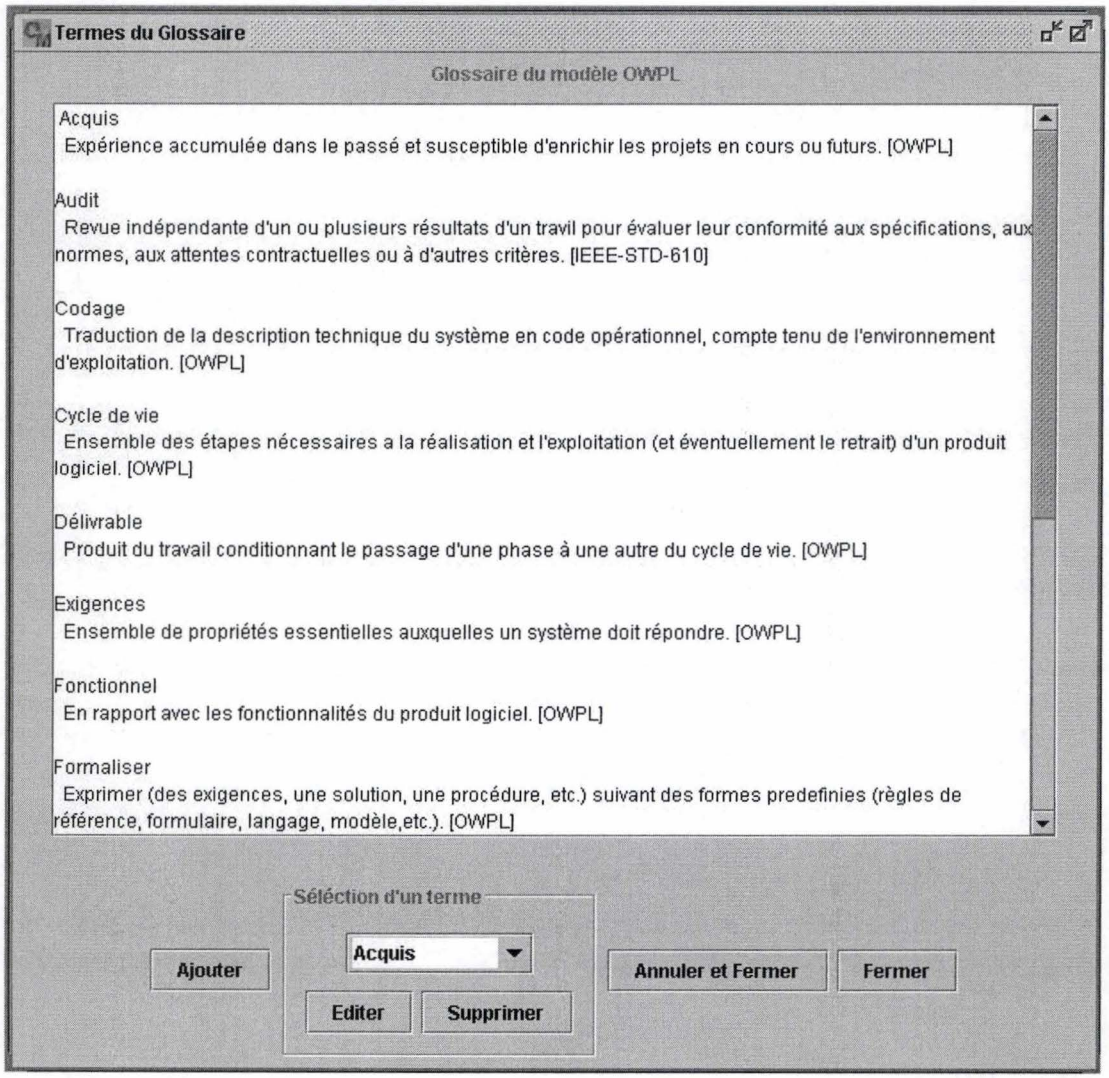


Figure 18: Vue du glossaire à partir de OWPL Manager.

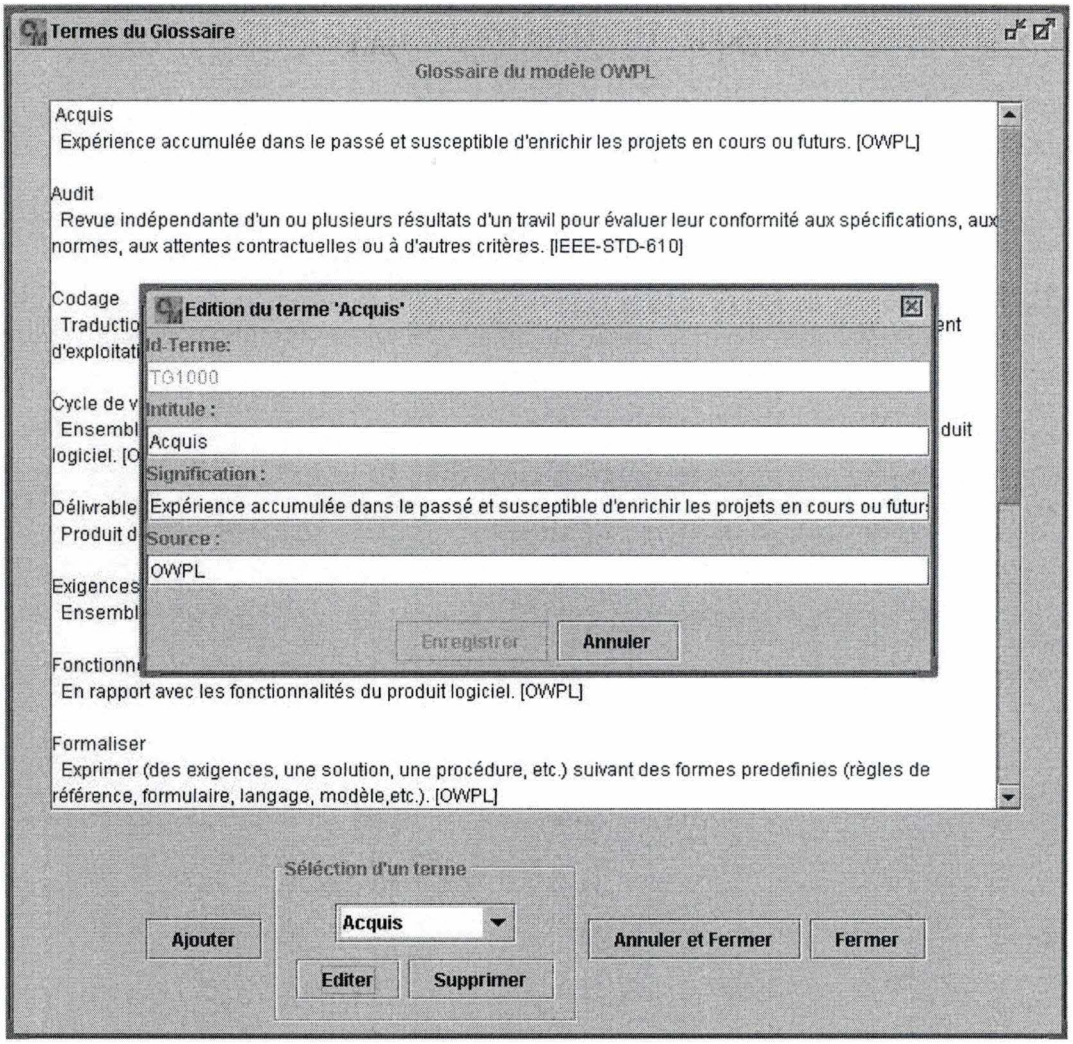


Figure 19: Edition d'un terme dans OWPL Manager

Toutes les fenêtres présentes ici n'apparaissent pas directement, d'autres n'apparaissent qu'après une question de confirmation. Représenter ici les interfaces de toutes ces transitions ajouterait de manière considérable le nombre d'illustrations.

Nous pouvons juste prendre l'exemple d'une petite fenêtre qui apparaît quand on veut ajouter une entrée à une pratique avant d'avoir la liste de tous les délivrables comme montré sur la figure 17. La fenêtre illustrée sur la figure 20 permet de préciser l'endroit d'insertion du délivrable sur la liste des délivrables. Une demande similaire est effectuée pour ajouter une pratique sur la liste des pratiques d'un processus, ou pour ajouter un processus dans le modèle OWPL.

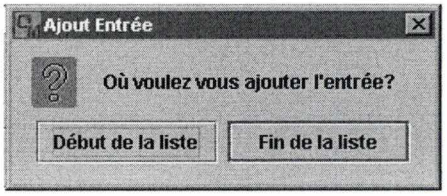


Figure 20: Demande de précision avant d'ajouter un délivrable à une pratique

6.1.3.2 Consultation du modèle OWPL

Pour rappel, l'application OWPL Manager, dans sa partie gestion, permet de générer une page web qu'on peut alors utiliser pour la consultation du modèle OWPL. On peut imaginer par exemple de mettre cette page dans l'intranet d'une entreprise afin que toutes les différentes personnes intéressées puissent prendre connaissance du modèle.

La page web est constitué de la page d'accueil illustrée par la figure 21. L'utilisateur peut alors utiliser la barre d'outils située à gauche pour accéder au modèle OWPL, au questionnaire, aux facteurs de succès, ou au glossaire.

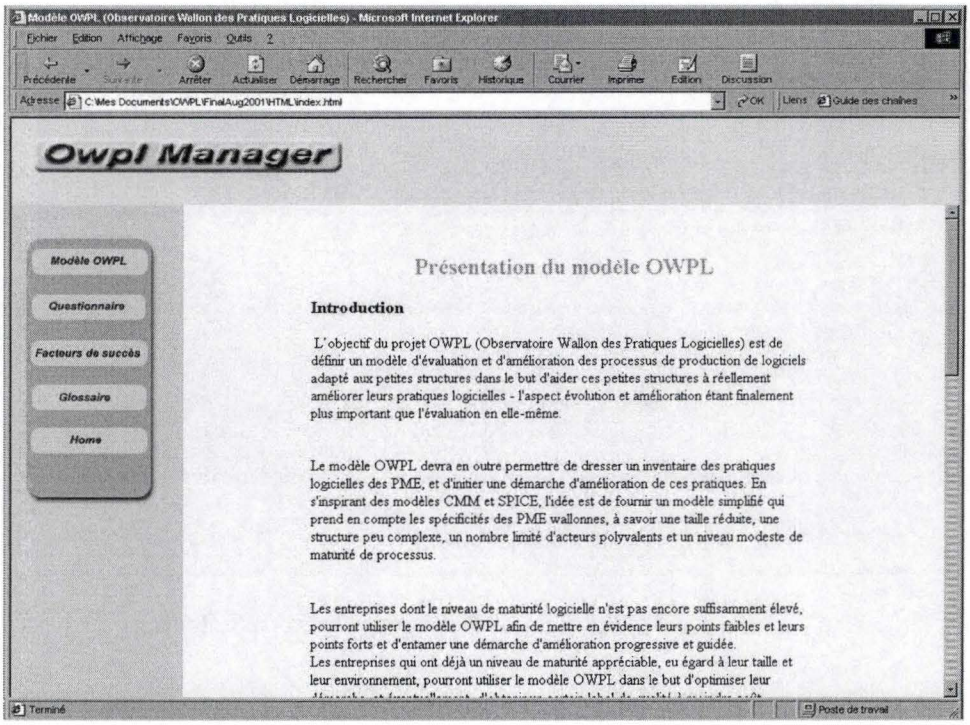


Figure 21: Page d'accueil de la présentation du modèle OWPL.

Après avoir accédé au modèle OWPL, l'utilisateur se trouve devant le page sur la figure 22. A partir de cette page il peut consulter tous les processus du modèle. A chaque processus est alors dédié une page web qui explique tut le contenu du processus. On a un exemple de consultation d'un processus sur la figure 23. Il s'agit ici de la consultation des détails du processus "Gestion des exigences".

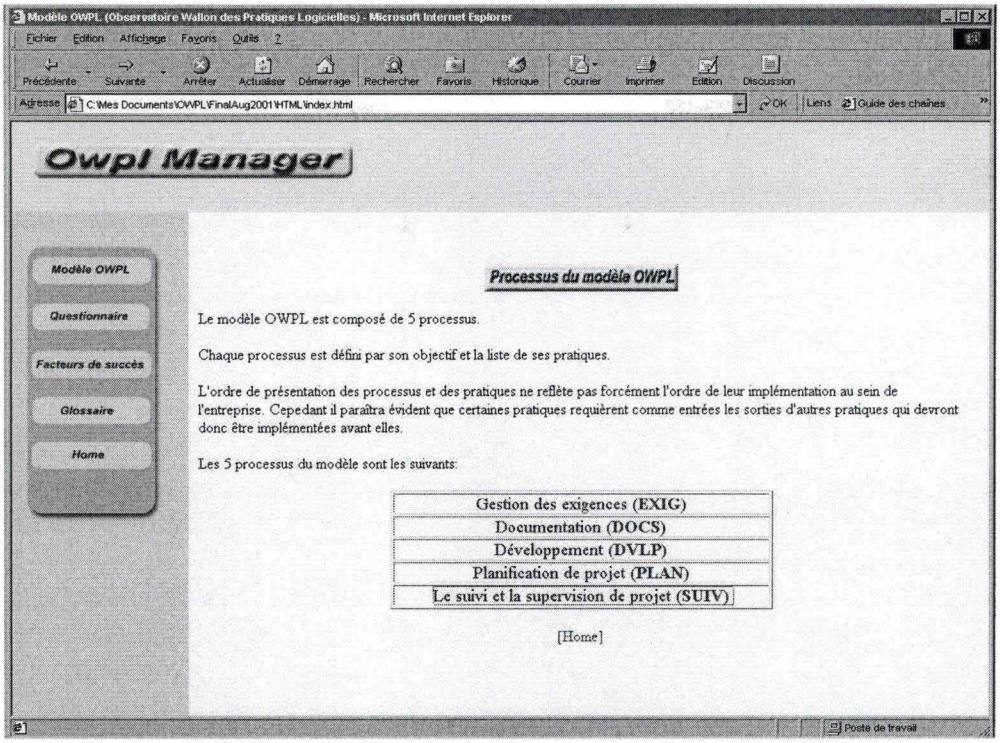


Figure 22: Consultation des processus du modèle OWPL

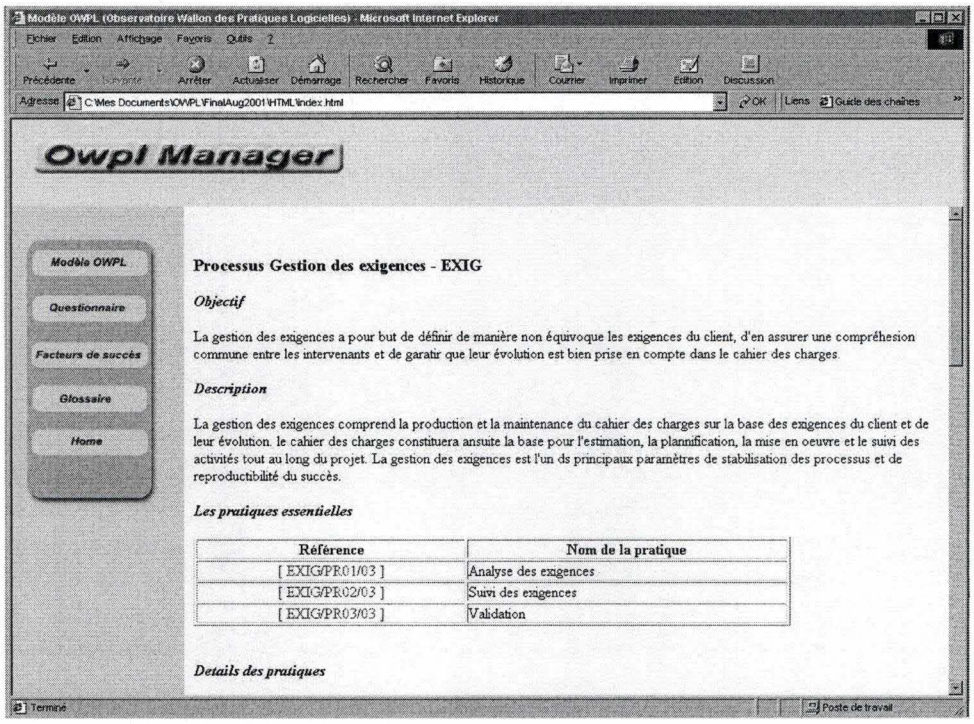


Figure 23: Détail d'un processus dans la consultation du modèle OWPL

Dans le cas de la consultation du glossaire, l'utilisateur se trouve devant la page illustrée par la figure 24 qui présente tous les termes présents dans le glossaire, avec au début de la page, une index qui renvoie aux lettres d'alphabet correspondantes, en vue de faciliter la navigation sur la liste triée de tous les termes du glossaire.

Comme nous l'avons déjà expliqué, les autres liens de la barre d'outils sont actuellement inactifs, leurs éléments correspondants n'ont pas encore été développés : il s'agit du questionnaire et des facteurs de succès.

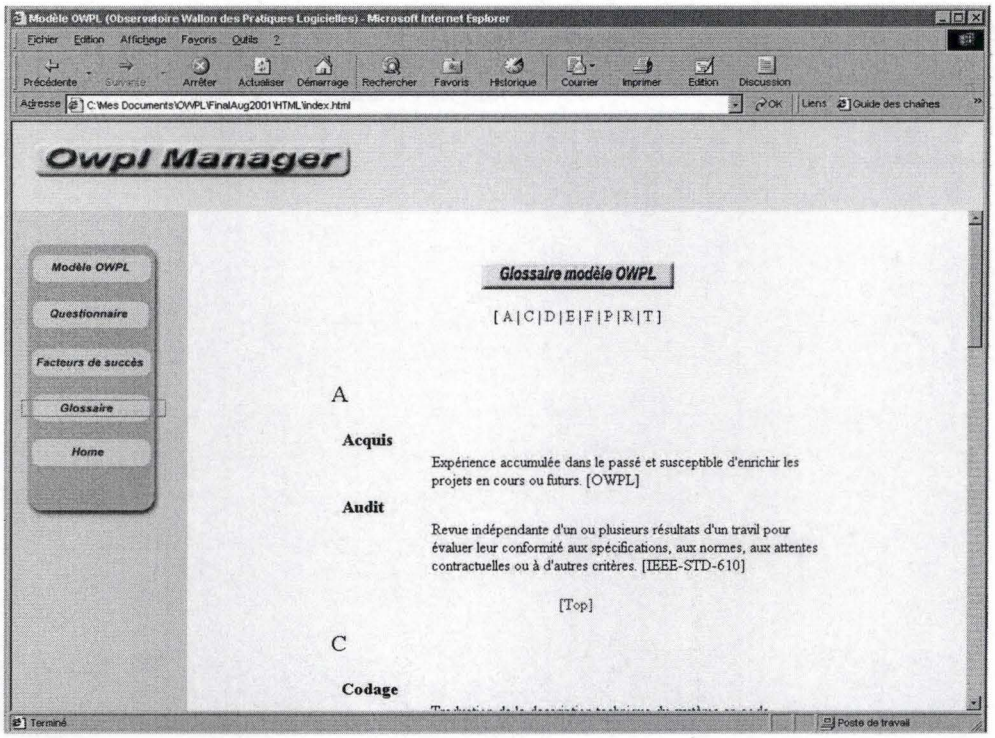


Figure 24: Consultation du glossaire du modèle OWPL.

IV. CONCLUSION GENERALE

Conclusion générale

Dans le cadre de ce mémoire, nous avons voulu proposer une application portable de gestion du modèle OWPL. C'était l'objectif que nous nous étions fixé, et après avoir pondéré plusieurs possibilités, nous avons opté pour une application nommée *OWPL Manager* implémentée en Java et qui utilise des fichiers XML comme base des données. Cette application génère automatiquement une page web sous forme de fichiers HTML qui présente tous les outils nécessaires à l'évaluation suivant le modèle OWPL.

Loin d'être la solution parfaite, l'application *OWPL Manager* offre l'avantage de pouvoir être utilisée avec plusieurs plates-formes de manière transparente pour l'utilisateur.
La partie développée peut déjà prétendre gérer le modèle OWPL de manière totalement automatique. C'est un apport majeur dans la mesure où toute mise à jour du modèle s'est toujours faite manuellement jusqu'ici. Seuls les questionnaires et les facteurs de succès restent à être développés avec les mêmes principes de programmation utilisés pour le modèle OWPL et le glossaire.

Malheureusement, on se rend vite compte que l'application *OWPL Manager* n'utilise qu'une infime partie des potentialités des documents XML. Elle se passe, par exemple, de toute la puissance des feuilles de style XSL qui, en elle même, peuvent fournir un ensemble de documents HTML décrivant le modèle à partir des documents XML.

Par ailleurs, le lecteur pourrait également se demander pourquoi nous n'avons pas utilisé de simples fichiers textes, car il n'y a pas une utilisation efficace des fichiers XML. La raison est simplement que notre application va évoluer avec l'implémentation des outils du langage XML.

En effet, peu de navigateurs et peu d'outils de développement prennent actuellement en charge le langage XML de manière à ce qu'une application XML tourne sans trop de problèmes. Nous l'avons vu aussi avec les différents parseurs XML qui sont implémentés. Même si les interfaces DOM et SAX sont censées être standards, leur implémentation souffre parfois de quelques particularités introduites par les développeurs. C'est un peu utopique, mais si nous arrivons un jour à une prise en charge de manière plus ou moins complète et standard de XML dans la majorité des navigateurs utilisés, notre application évoluerait vers une application qui se réduirait à de la manipulation de fichiers XML à travers un parseur. La partie consultation serait complètement supportée par des feuilles de style destinées à l'ensemble des fichiers XML.

Nous avons également vu que la signature des applets Java a constitué un frein pour l'utilisation des applets Java. La plus grande faiblesse est la différence entre les méthodes utilisées à cette fin par les navigateurs. C'est dommage, car cela aurait pu aboutir à une application sur une interface très familière à l'utilisateur. Dans ce cas, aussi bien la gestion du modèle que la consultation se seraient faites à partir du navigateur. Evidemment, il est impossible d'arriver à une harmonisation des méthodes dans ce domaine, car la notion même de sécurité nécessite une diversification des méthodes pour tout une meilleur efficacité.

Au stade actuel de développement, l'ensemble des interfaces découle simplement du scénario modélisé par le diagramme de flux. Or, pour une application de la taille de celle-ci, il importe qu'une démarche spécifique et rigoureuse de détermination de l'interface homme machine soit effectuée. Cette démarche peut être menée avec la méthode TRIDENT[22].

Plusieurs autres parties à implémenter dépendent du développement du modèle OWPL lui-même. Nous pouvons citer l'exemple de l'utilisation du glossaire dans l'application et le lien entre les facteurs de succès et les pratiques.

Pour le premier, il est prévu dans nos exigences de départ que toute utilisation d'un terme présent dans le glossaire offre la possibilité d'utiliser la signification présente dans le glossaire. Il s'agit en fait de l'utilisation dynamique du glossaire dans l'application. Or, il se fait qu'actuellement, il n'y a pas de règle précise qui régit la présence d'un terme dans le glossaire. C'est ce qui fait qu'on ne sait pas quand est-ce qu'il faut utiliser une signification dynamique (faire référence à la définition du glossaire), et quand est-ce qu'il ne faut pas le faire, c'est-à-dire associer directement les termes à leur signification sur les fichiers XML, comme nous l'avons fait dans notre application. Il va de soi qu'une fois des règles précises seront établies, on pourra utiliser des significations dynamiques. L'avantage sera qu'il n'y aura plus de redondance dans la signification des termes, et toute modification unique d'une signification aurait le même effet sur l'ensemble du modèle.

Pour le second, il s'agit de préciser le lien qui peut exister entre des facteurs de succès et des pratiques. Actuellement, le modèle OWPL prévoit que tous les facteurs de succès supportent toutes les pratiques. Or, l'utilisation fréquente du modèle OWPL présage qu'il pourrait exister des liens entre certains facteurs de succès et certaines pratiques. C'est une relation qui doit être clairement définie avant que cet aspect soit pris en charge dans notre modèle.

Aussi bien pour le premier cas que pour le second, l'application actuelle est conçue de manière à prendre en charge toute évolution du modèle OWPL dans ces sens.

Les versions futures de l'application *OWPL Manager* devront également prendre en charge l'exportation de divers documents imprimables qui ressemblent au style utilisé pour les différent délivrables ayant servi pour la publication du modèle OWPL. Pour ce faire, plusieurs technologies peuvent être utilisées. On peut par exemple penser à l'utilisation des fichiers

[22] L'objectif du projet TRIDENT des FUNDP est de créer une méthode de conception et de développement des interfaces Homme-Machine destinées à des application de gestion hautement interactives.
Les caractéristiques de la méthode TRIDENT sont les suivantes :

- l'analyse de la tâche constitue la source du processus de conception ;
- la conception de l'interface est structurée à partir de trois composants relativement autonomes : la présentation, le dialogue et les services de l'application.
- l'architecture du logiciel repose sur ces trois composants :
  1. Les spécifications ergonomiques de la présentation et du dialogue sont dérivées de l'analyse de la tâche ;
  2. Des outils logiciels supportent la conception et le développement de l'Interface. En particulier, pour les interfaces de type formulaire, la présentation est générée à partir d'un ensemble de règles ergonomiques sélectionnées au terme de l'analyse de la tâche ;
  3. La méthode TRIDENT est intégrable dans une méthode générale de développement de systèmes d'information.

Les recherches actuelles du projet TRIDENT concernent l'ingénierie de l'analyse de la tâche et l'ingénierie de l'analyse de la conception et du développement des dialogues.

RTF ou des fichiers Latex. Dans les deux cas, il s'agit d'un format de fichier basé sur l'utilisation des balises particuliers. Il faudrait évidemment aussi veiller à utiliser une technologie portable, et c'est le cas pour les deux technologies citées ici.

En conclusion, nous pouvons dire que la partie développée jusqu'ici a encore du chemin à faire avant l'utilisation effective et complète de l'application. Elle offre néanmoins l'avantage d'avoir jeté les bases du développement d'une application portable de gestion du modèle OWPL et présente une grande flexibilité face à des améliorations futures au regard des potentialités que présente l'évolution des outils XML.

V. BIBLIOGRAPHIE

[1] HABRA N., NIYITUGABIRA E. ET RENAULT A., *Modèle OWPL : Evaluation et Amélioration des Pratiques Logicielles dans les PME Wallonnes*, Technical Report 1/99, OWPL-FUNDP,1999.

[2] LOBET-MARIS CI., Utilisation des systèmes d'information interorganisationnels par les PME belges – Rapport Final – CITA – FUNDP – novembre 1997.

[3] http://www.alcyonix.com/fr/faq/spice.htm

[4] http://www.alcyonix.com/fr/faq/cmm.htm

[5] HAINAUT J-L., *Base de données et modèles de calcul, Outils et méthodes pour l'utilisateur*, InterEditions, Paris, 1994.

[6] HABRA N., ET RENAULT A., *Modèle OWPL : Questionnaire, version 1.2.3,* OWPL-FUNDP, 2000.

[7] HAROLD E.R., *XML le guide de l'utilisateur*, Osman Euyrolles multimedia, Février 2000.

[8] LIBERTY J., *XML*, Campuspress, 2000.

[9] KOLBECK R.0, *Grand Livre Javascript*, Micro application, 1997.

[10] HABRA N., NIYITUGABIRA E. & RENAULT A., *Modèle OWPL - Glossaire*, OWPL-FUNDP 1999

[11] HABRA N. & RENAULT A., *Modèle OWPL - Méthode de cotation*, OWPL-FUNDP 2000

[12] http://www.citeweb.net/apetitje/xml/index.html

[13] http://tornade.ere.umontreal.ca/~marcoux/ottawa/marcoux.html

[14] http://www.chez.com/xml

[15] http://docsdunet.free.fr/doc_xml.html

[16] http://tulipe.cnam.fr/~farinone/IHM/CoursXML.html

[17] http://www.pourlascience.com/numeros/pls-261/art-4.htm

[18] ANDREW TANENBAUM, *Réseaux 3ème édition*, Prentice Hall International, DUNOD, 1999.

[19] GARY CORNELL, CAY S. HORSTMANN, *Au cœur de Java 2*, Campuspress, 2000.

[20] GARY CORNELL, CAY S. HORSTMANN, *Au cœur de Java 2, fonctions avancées*, Campuspress, 2000.

[21] A. MENEZES, P. OORSCHOT, S. VANSTONE, *Handbook of Applied cryptography*, CRC Press, 1997.

[22] http://www.securingjava.com/appdx-c/appdx-c-1.html

[23] http://developper.java.sun

[24] A. Menezes, P. Oorschot, S. Vanstone, *Handbook of Applied cryptography*, CRC Press, 1997.

[25] F. BODART, Y. PIGNEUR, *Conception assistée des systèmes d'informations – Méthode, modèles, outils*, Masson, 1989.

VI. ANNEXES

Document annexe n°1

Le modèle OWPL version 1.2.2 b

Modèle OWPL

Evaluation et amélioration des pratiques logicielles dans les PME wallonnes

Version 1.2.2 b

N. Habra, A. Renault

Institut d'Informatique
FUNDP - Charleroi

OWPL - Modèle public .doc, version du 10/01/01

Table des matières

1. Introduction

L'objectif du projet *OWPL* (Observatoire Wallon des Pratiques Logicielles) est de définir un modèle d'évaluation et d'amélioration des processus de production de logiciels adapté aux petites structures dans le but d'aider ces petites structures à réellement améliorer leurs pratiques logicielles - l'aspect évolution et amélioration étant finalement plus important que l'évaluation en elle-même.

Le modèle *OWPL* devra en outre permettre de dresser un inventaire des pratiques logicielles des PME, et d'initier une démarche d'amélioration de ces pratiques. En s'inspirant des modèles CMM et SPICE, l'idée est de fournir un modèle simplifié qui prend en compte les spécificités des PME wallonnes, à savoir une taille réduite, une structure peu complexe, un nombre limité d'acteurs polyvalents et un niveau modeste de maturité de processus.

Les entreprises dont le niveau de maturité logicielle n'est pas encore suffisamment élevé, pourront utiliser le modèle *OWPL* afin de mettre en évidence leurs points faibles et leurs points forts et d'entamer une démarche d'amélioration progressive et guidée.

Les entreprises qui ont déjà un niveau de maturité appréciable, eu égard à leur taille et leur environnement, pourront utiliser le modèle *OWPL* dans le but d'optimiser leur démarche, et éventuellement, d'obtenir un certain label de qualité à moindre coût.

Le modèle *OWPL* devra contribuer à améliorer le processus de production de logiciels dans les entreprises wallonnes. Il devra en outre contribuer, par l'effet de sensibilisation qu'il crée, à améliorer l'image de la production du software dans l'industrie en général. Cela contribuera à la création d'une industrie innovante et performante.

Le document est structuré en quatre parties. Après cette introduction, la section 2 décrit de façon générale la logique sous-jacente à l'élaboration du modèle. La section 3 donne une présentation détaillée du modèle et la section 4 décrit la méthodologie à suivre pour son utilisation dans le cadre d'une démarche d'évaluation et d'amélioration des pratiques logicielles en entreprise.

Ce document donne une version allégée du modèle. Le lecteur désireux d'approfondir le sujet pourra trouver la description complète des pratiques et des facteurs de succès dans le document d'origine[1].

[1] Ce document est disponible auprès du Laboratoire de Qualité Logicielle, FUNDP - CTTC, Boulevard Tirou 130, B-6000 Charleroi, Belgique. E-mail : owpl@info.fundp.ac.be

2. La démarche de conception du modèle OWPL

La taille et la complexité des modèles comme CMM et SPICE rendent leur implémentation excessivement lourde et coûteuse pour des PME. Le volume d'information qu'ils contiennent et la quantité de processus et d'attributs qui y sont définis les rendent inutilisables par des petites structures. D'autre part, le recours à une société spécialisée dans l'amélioration des pratiques logicielles est excessivement coûteux. Selon une étude[xviii] de 1997, le coût d'une telle intervention représente 50% du budget annuel de près de 42% des PME wallonnes. La mise en œuvre d'une telle démarche, outre l'investissement financier, implique la mise à disposition de personnes et de ressources. Or, selon la même étude, 64% des PME planifient leur activité principale de manière journalière, et 22% de manière hebdomadaire, et dans près de deux entreprises sur trois les technologies de l'information ne font pas l'objet d'une prise en charge formalisée dans l'organigramme de l'entreprise.

La simplicité des structures des entreprises wallonnes, la petite taille des équipes logicielles et leur mode de fonctionnement justifiaient donc la nécessité d'une adaptation des modèles existants à leurs particularités.

Le modèle *OWPL* est conçu de façon à permettre la mise en évidence rapide des pratiques à améliorer, l'établissement d'un plan d'actions simple visant à améliorer ces pratiques et la prise de mesures simples de la variance induite.

2.1. Contraintes d'adaptation

La démarche suivie pour la définition de la structure du modèle *OWPL* et de la méthode d'évaluation s'est inspirée des conclusions des deux études réalisées par ailleurs se rapportant à l'utilisation de CMM[xix] et de SPICE[xxiii] dans des structures informatiques de petite taille.

Pour rappel, ces conclusions étaient globalement les suivantes :

- Nécessité de tenir compte du contexte particulier des PME wallonnes en mettant l'accent sur les aspects "évolution" et "amélioration", l'attribution d'un label de qualité n'est pas prioritaire ;
- Nécessité d'utiliser un vocabulaire accessible et non équivoque et de ne laisser aucune ambiguïté d'interprétation dans les conclusions des évaluations ;
- Nécessité d'un encadrement plus important pour sensibiliser les PME aux objectifs de la démarche d'amélioration, et limiter ainsi l'investissement qu'elles devront consentir ;
- Nécessité de définir une méthodologie d'utilisation du modèle incluant notamment l'étape de sensibilisation et un système de communication efficace pour la diffusion des résultats ;
- Inutilité de s'attaquer aux processus des niveaux de maturité supérieurs, surtout dans un premier temps ;
- Nécessité d'éviter la bureaucratie et de limiter au strict minimum le nombre de documents ;
- Nécessité de définir les responsabilités en insistant davantage sur la réalisation effective des tâches que sur la répartition de celles-ci entre les différentes personnes.

Une première expérience sur le terrain dans le cadre *de micro-évaluations des pratiques logicielles*[xxiv] nous a également permis d'orienter notre démarche sur des bases concrètes.

2.2. Contexte d'utilisation du modèle OWPL

La pratique montre que de nombreuses entreprises ont des difficultés à préciser les objectifs à atteindre dans la démarche d'amélioration de leurs pratiques logicielles. Partant de cette constatation, nous avons jugé important de définir l'environnement dans lequel une telle démarche allait devoir se produire, de même que l'environnement dans lequel la démarche initiale de production de logiciel se produit.

Le modèle *OWPL* repose sur l'hypothèse que toute activité d'une entreprise doit être effectuée dans l'optique de la réalisation des objectifs de l'entreprise. Chaque activité aura son objectif propre et concourra à la réalisation de l'objectif du processus auquel elle appartient. Elle apporte ainsi sa contribution à la réalisation de l'objectif du processus qui est lui-même défini en fonction de l'objectif global du département (Ex. : les processus logiciels). Cet objectif est à son tour défini eu égard à l'objectif général de l'entreprise. Les objectifs organisés de cette manière sont donc toujours en accord avec les objectifs de l'entreprise. (Cfr. Figure 1 : L'arbre des objectifs).

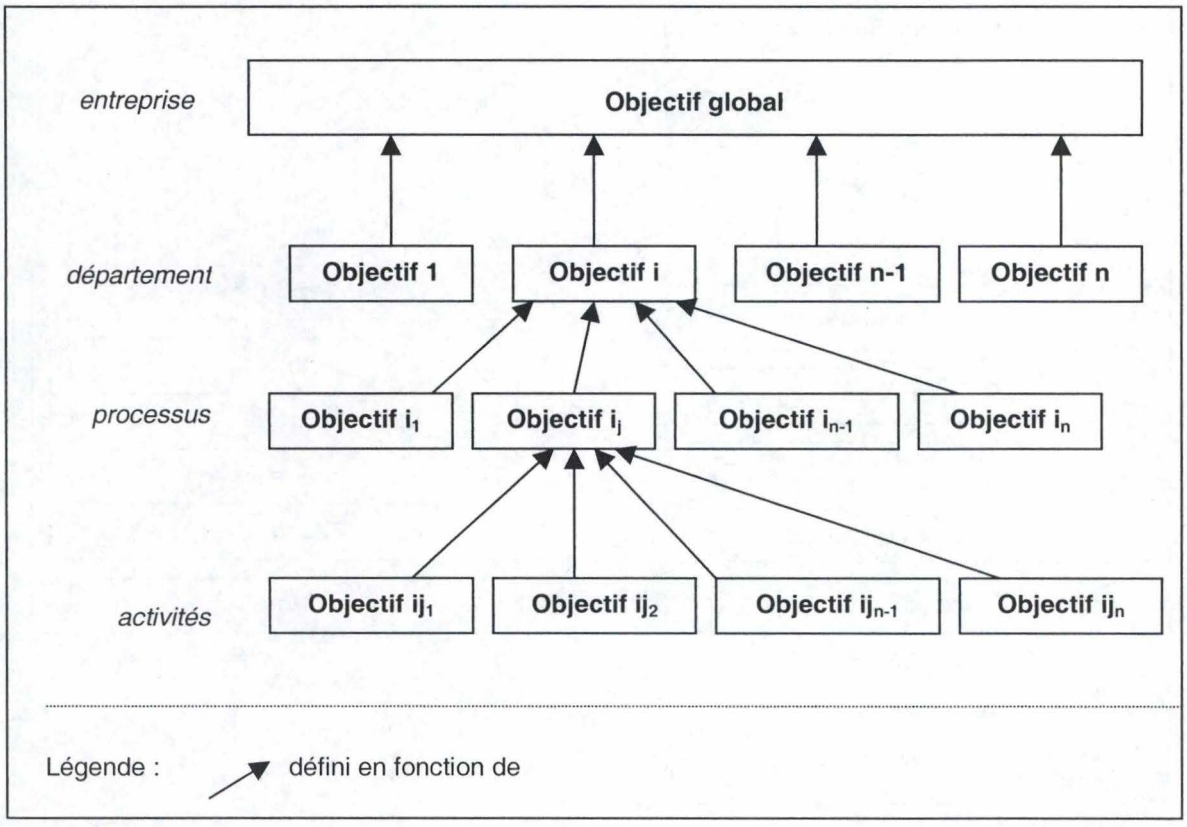


Figure 1 : L'arbre des objectifs

La définition des objectifs est essentielle dans toute démarche d'amélioration. Elle permet notamment de cibler l'effort, de motiver les acteurs et de contrôler l'efficacité de la démarche.

En outre, un certain nombre de facteurs doivent être réunis afin de stabiliser l'environnement d'exécution de processus et garantir le succès. Ce sont des facteurs de succès. On citera notamment la mise à disposition des ressources, l'engagement de la direction, une organisation efficace, etc.

2.3. Structure du modèle OWPL

L'élément central du modèle *OWPL* est le **processus.**
Le processus est organisé de manière à répondre à une préoccupation de l'entreprise, et défini en fonction d'un objectif qui contribue à la réalisation de l'objectif global de l'entreprise. Chaque processus est composé de **pratiques** nécessaires pour sa mise en œuvre effective. Il se déroule dans un environnement où **des facteurs de succès** garantissent sa performance.

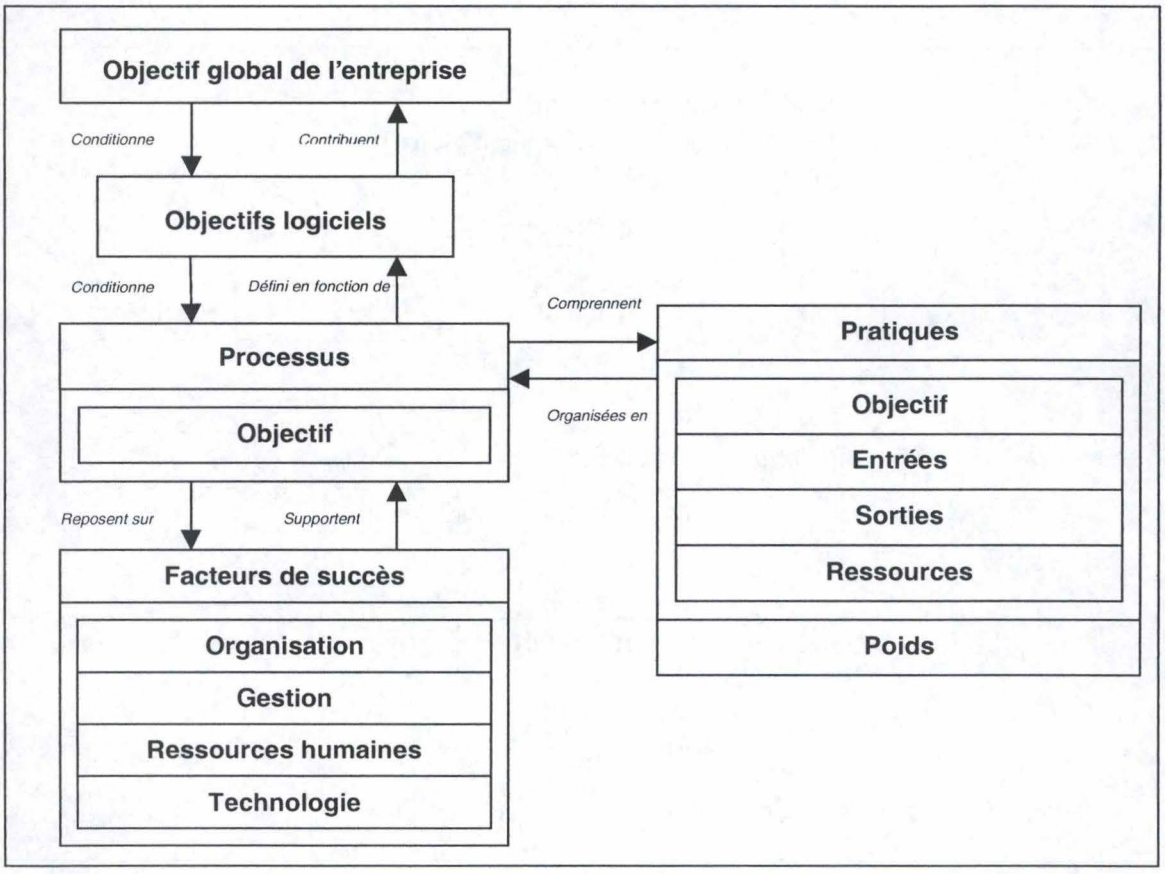


Figure 2 : Structure du modèle OWPL

2.3.1 Le processus

Un processus est un ***ensemble structuré de pratiques nécessaires à la réalisation d'un objectif commun clairement défini.***

Le processus est défini en fonction de l'objectif global de l'entreprise. S'appuyant sur des *facteurs de succès*, il permet d'organiser différentes pratiques en un tout cohérent et contribue ainsi à la réalisation de l'objectif de l'entreprise.

Chaque processus est défini par son objectif et la liste des pratiques qui le composent. (Cfr. Figure 2 : Structure du modèle *OWPL*)

2.3.2 Les pratiques

Une pratique est une ***activité d'ingénierie qui contribue à la réalisation de l'objectif d'un processus par la création d'un délivrable ou l'amélioration de la capacité du processus.*** Chaque pratique est caractérisée par son objectif, ses entrées, ses sorties, ses ressources et son poids.

Les pratiques sont organisées de façon à faciliter leur articulation autour d'un objectif central, celui du processus dont elles font partie. Chaque pratique est décrite par 5 attributs : son objectif, ses entrées, ses sorties, les ressources qui lui sont nécessaires et son poids dans la réalisation de l'objectif du processus auquel elle appartient. (Cfr. Figure 2 : Structure du modèle *OWPL*)

2.3.3 Les facteurs de succès

Les facteurs de succès sont des ***éléments d'environnement qui favorisent la mise en place d'un support permettant une exécution optimale des processus.***

Ils sont regroupés en quatre catégories : l'**organisation** dans laquelle se déroulent les processus, la politique de **gestion** mise en place, les **ressources humaines** mobilisées et les moyens **techniques** utilisés. (Cfr. Figure 2 : Structure du modèle *OWPL*)

3. Présentation détaillée du modèle OWPL

3.1. Les processus

Chaque processus est défini par son objectif et la liste de ses pratiques.

L'ordre de présentation des processus et des pratiques ne reflète pas forcément l'ordre de leur implémentation au sein de l'entreprise. Cependant, il paraîtra évident que certaines pratiques requièrent comme entrées les sorties d'autres pratiques qui devront donc être implémentées avant elles.

Les modèle *OWPL* est composé des 10 processus suivants :

- La gestion des exigences **(EXIG)**
- La documentation **(DOCS)**
- La planification de projet **(PLAN)**
- Le suivi et la supervision de projet **(SUIV)**
- Le développement **(DVLP)**
- Les tests **(TEST)**
- La gestion de configuration **(CONF)**
- La gestion des sous-traitants **(SSTR)**
- La gestion de la qualité **(QUAL)**
- La capitalisation des acquis **(CPTL)**

3.1.1 La gestion des exigences - EXIG

Objectif

La **gestion des exigences** a pour but de définir de manière non équivoque les exigences du client, d'en assurer une compréhension commune entre les intervenants et de garantir que leur évolution est bien prise en compte dans le cahier des charges.

Description

La gestion des exigences comprend la production et la maintenance du cahier des charges sur base des exigences du client et de leur évolution. Le cahier des charges constituera ensuite la base pour l'estimation, la planification, la mise en œuvre et le suivi des activités tout au long du projet.
La gestion des exigences est l'un des principaux paramètres de stabilisation des processus et de reproductibilité du succès.

Les pratiques essentielles

Référence	Nom de la pratique
[EXIG/PR01/03]	Analyse des exigences
[EXIG/PR02/03]	Suivi des exigences
[EXIG/PR03/03]	Validation

3.1.2 La documentation - DOCS

Objectif

La **documentation** a pour but de développer et maintenir les documents nécessaires pour installer, utiliser, et supporter efficacement le produit logiciel.

Description

Les documents concernés sont décrits dans le cahier des charges. Leur contenu sera donc amené à évoluer en fonction d'éventuelles modifications des exigences en cours de projet.
Sont notamment concernés les documents de présentation générale, les manuels de référence, les supports pour les formations, les manuels d'installation et d'utilisation.
La gestion des documents produits par les autres processus n'est pas concernée ici, mais est prise en compte par le processus de gestion de la configuration.

Les pratiques essentielles

Référence	Nom de la pratique
[DOCS/PR01/03]	Identification des exigences en terme de documentation
[DOCS/PR02/03]	Développement des documents
[DOCS/PR03/03]	Mise à disposition des documents aux personnes concernées

3.1.3 La planification de projet - PLAN

Objectif

La **planification de projet** a pour but d'établir des prévisions raisonnables sur les ressources nécessaires pour la mise en œuvre des tâches de développement et de gestion de projet et d'attribuer ces ressources à ces tâches.

Description

La planification du projet permet au chef de projet d'organiser le travail de son équipe sur base de données objectives : les tâches à effectuer en interne et celles à sous-traiter, les ressources nécessaires et les ressources disponibles (ressources de temps, ressources financières, ressources humaines, ressources techniques), et l'avancement réel du projet en cours. Le planning permet d'avoir une vue d'ensemble sur le projet, et de mettre en évidence le risque en tenant compte des contraintes associées à chaque tâche.
Les estimations de la taille des différents produits, des ressources nécessaires et des risques associés à ces ressources sont calculées en tenant compte des expériences antérieures.

La planification est le mécanisme de base de contrôle et de gestion du projet. Les prévisions qu'elle permet d'établir sont cruciales pour une bonne gestion du projet. Elle permet notamment un suivi efficace des activités du projet et elle permet de mettre en évidence les difficultés éventuelles à respecter les engagements.

Les pratiques essentielles

Référence	Nom de la pratique
[PLAN/PR01/06]	**Découpage du projet en tâches**
[PLAN/PR02/06]	**Identification des contraintes associées aux tâches**
[PLAN/PR03/06]	**Estimation des ressources nécessaires**
[PLAN/PR04/06]	**Analyse du risque**
[PLAN/PR05/06]	**Elaboration du planning**
[PLAN/PR06/06]	**Adaptation du planning en fonction de l'évolution du contexte**

3.1.4 Le suivi et la supervision de projet - SUIV

Objectif

Le **suivi et la supervision de projet** ont pour but de s'assurer que le projet se déroule conformément aux prévisions du planning. Ce processus suit de façon objective et précise l'avancement du projet pour déterminer les écarts éventuels par rapport aux prévisions et, le cas échéant, prendre les mesures correctives qui s'imposent.

Description

Ce processus consiste à évaluer en permanence l'avancement réel du projet, les ressources utilisées, les écarts entre ce qui a été planifié et ce qui a effectivement été réalisé, et enfin la quantité de travail restant à effectuer.
Le suivi doit être objectif et précis pour que les résultats soient exploitables. Ceux-ci doivent permettre de déterminer les origines des écarts constatés et de prendre des mesures adéquates afin de résorber et de prévenir ces écarts.

Les pratiques de suivi et de supervision de projet permettent de gagner de la visibilité sur les activités du projet. Les difficultés en matière de respect des engagements sont identifiées au fur et à mesure qu'elles surgissent et les risques de crise majeure sont atténués.

Les pratiques essentielles

Référence	Nom de la pratique
[SUIV/PR01/04]	Enregistrement de l'avancement du projet
[SUIV/PR02/04]	Analyse de l'avancement du projet
[SUIV/PR03/04]	Prise de décisions correctives
[SUIV/PR04/04]	Clôture et bilan du projet

3.1.5 Le développement - DVLP

Objectif

Le processus de développement englobe toutes les étapes liées à la production du produit logiciel proprement dite, c'est-à-dire l'**analyse fonctionnelle**, la **conception**, le **codage,** le **déboguage**, la **mise en exploitation** et la **maintenance**.

Description

Le processus de développement intègre et exécute les pratiques de développement de façon cohérente pour produire le logiciel. Le processus de développement débute par la description des fonctions du logiciel sur base des exigences du client. De cette description sera dégagée l'architecture du système et une description technique de chacun des composants qui seront alors traduits en code opérationnel. Le logiciel ainsi produit sera mis en exploitation après suppression des erreurs et validation par le client. On considère que les **corrections** apportées à la solution après sa **mise en exploitation** font également partie du processus de développement.

Les pratiques essentielles

Référence	Nom de la pratique
[DVLP/PR01/05]	Analyse fonctionnelle
[DVLP/PR02/05]	Conception
[DVLP/PR03/05]	Codage
[DVLP/PR04/05]	Déboguage
[DVLP/PR05/05]	Mise en exploitation et maintenance corrective

3.1.6 Les tests - TEST

Objectif

Le processus de **tests** a pour but de vérifier l'adéquation du produit logiciel par rapport aux exigences, et de détecter les erreurs.

Description

Les plans de tests sont définis dès le début du projet, et affinés à chaque phase. Ils doivent permettre de vérifier qu'un produit logiciel est conforme aux exigences en tenant compte de la criticité du produit et des critères de qualité prédéfinis.

Les pratiques essentielles

Référence	Nom de la pratique
[TEST/PR01/04]	Définition de la stratégie de tests
[TEST/PR02/04]	Définition des jeux de tests
[TEST/PR03/04]	Suivi des tests
[TEST/PR04/04]	Conduite des tests

3.1.7 La gestion de configuration - CONF

Objectif

La **gestion de configuration** a pour but d'établir et de maintenir la cohérence entre les composants du projet tout au long de son *Cycle de Vie*. Elle consiste à identifier la configuration du produit logiciel à des moments déterminés, à contrôler systématiquement les changements apportés à cette configuration, et à maintenir son *intégrité* et sa *traçabilité*.

Description

La gestion de configuration permet non seulement de déterminer tous les éléments de configuration d'un produit logiciel, mais également d'en identifier les différentes versions, de déterminer les relations entre différentes applications au niveau du partage des librairies ou de l'accès aux bases de données, de déterminer les applications qui sont toujours en phase de test, etc.
Sont concernés, d'une part, les produits livrés au client et les éléments nécessaires à leur création ou identifiés avec eux (sources, structure de données, documentation, exécutables, versions de matériel, etc.) et, d'autre part, tous les documents produits dans le cadre de la gestion de projet.

Les pratiques essentielles

Référence	Nom de la pratique
[CONF/PR01/03]	Identification des produits du travail
[CONF/PR02/03]	Identification des relations entre les produits du travail
[CONF/PR03/03]	Suivi de la configuration

3.1.8 La gestion des sous-traitants - SSTR

Objectif

La **gestion des sous-traitants** a pour but de définir les engagements réciproques du donneur d'ordres et du sous-traitant, et de mettre en place la structure qui permet d'assurer le respect des exigences du cahier des charges de sous-traitance.

Description

La maîtrise du processus de gestion de la sous-traitance permet à une organisation d'intégrer dans ses plans les activités placées sous la responsabilité d'un tiers, et de suivre voire contrôler leurs performances vis-à-vis des engagements. L'organisation peut, de cette manière, appliquer ses processus au sous-traitant et garantir l'intégration du travail du sous-traitant dans le respect de la qualité du produit final.
Cette gestion permet de rendre transparentes les interventions du sous-traitant dans le processus de production de logiciel, ce qui permet de limiter les risques liés aux relations client/fournisseur et de favoriser la bonne fin des projets.

Les pratiques essentielles

Référence	Nom de la pratique
[SSTR/PR01/04]	Rédaction de l'appel d'offres de sous-traitance
[SSTR/PR02/04]	Sélection du sous-traitant
[SSTR/PR03/04]	Suivi de l'accord de sous-traitance
[SSTR/PR04/04]	Réception du travail sous-traité

3.1.9 La gestion de la qualité - QUAL

Objectif

La **gestion de la qualité** a pour but de s'assurer que le produit logiciel répond aux exigences de qualité de la société, et que les normes de qualité de tous les processus ont bien été respectées.

Description

La gestion de la qualité permet d'avoir une visibilité adéquate sur les processus mis en œuvre par le projet et sur les produits en élaboration. Le résultat du projet et la reproductibilité des processus sont assurés par le contrôle du respect des procédures définies pour chaque processus. La reproductibilité du processus assure de pouvoir disposer systématiquement de produits de qualité homogène, ce qui contribue à renforcer la confiance et la satisfaction du client.

Les pratiques essentielles

Référence	Nom de la pratique
[QUAL/PR01/05]	Définition/adaptation des normes et procédures
[QUAL/PR02/05]	Suivi de la qualité du produit
[QUAL/PR03/05]	Suivi de la qualité des processus
[QUAL/PR04/05]	Prise de décisions correctives
[QUAL/PR05/05]	Diffusion de l'information

3.1.10 La capitalisation des acquis - CPTL

Objectif

La **capitalisation des acquis** a pour but de généraliser à tous les projets (en cours ou futurs) les pratiques qui ont favorisé le succès des projets antérieurs.

Description

Les pratiques qui ont favorisé le succès des projets antérieurs sont identifiées lors de l'analyse des bilans de clôture des projets. Elles sont rassemblées afin de créer un cadre d'apprentissage qui permettra aux nouveaux projets de bénéficier de l'expérience des projets antérieurs.
De même, les pratiques qui ont pu être à la source de l'échec de projets antérieurs sont identifiées et écartées ou adaptées en conséquence.

Les pratiques essentielles

Référence	Nom de la pratique
[CPTL/PR01/03]	Analyse des projets antérieurs
[CPTL/PR02/03]	Définition/adaptation du cadre d'apprentissage
[CPTL/PR03/03]	Information continue sur le manuel qualité

3.2. Les pratiques

Une pratique est une ***activité d'ingénierie qui contribue à la réalisation de l'objectif d'un processus par la création d'un délivrable ou l'amélioration de la capacité du processus***.

Un processus est composé de plusieurs pratiques, mais une pratique n'est définie qu'au sein d'un seul processus. Cependant, l'accomplissement d'une pratique peut servir de préalable ou de gage de succès aux pratiques définies dans le même processus, voire dans d'autres processus.

Chaque pratique est définie par son objectif, ses entrées, ses sorties, les ressources qui lui sont nécessaires, et son poids dans la réalisation de l'objectif du processus auquel elle appartient.

L'objectif d'une pratique
Description synthétique de la portée, des limites et de l'intention de la pratique. La réalisation effective de la pratique doit concourir à atteindre l'objectif du processus auquel elle appartient.

Les entrées d'une pratique
Eléments utiles ou indispensables à la réalisation d'une pratique. Ces éléments sont généralement produits par d'autres pratiques. La liste des entrées proposées pour chaque pratique n'est pas exhaustive.

Les sorties d'une pratique
Eléments concrets attendus d'une pratique. Leur existence confirme la bonne exécution de la pratique. La liste des sorties proposées pour chaque pratique n'est pas exhaustive.

Les ressources d'une pratique
Les ressources représentent les éléments techniques, documents, outils ou méthodes qui aident à la réalisation de la pratique. La liste des ressources proposées pour chaque pratique n'est pas exhaustive.

Le poids d'une pratique
Une valeur intrinsèque (subjective) attribuée à la pratique pour souligner son importance dans la réalisation de l'objectif du processus auquel elle appartient.

Remarque : Le poids des pratiques étant lié directement à l'utilisation du questionnaire, leur description a été déplacée en conséquence.

Répartition des pratiques par processus
Le tableau suivant donne la répartition des pratiques par processus du modèle *OWPL* :

Processus	Nombre de pratiques
La gestion des exigences - EXIG	3
La documentation - DOCS	3
La planification de projet - PLAN	6
Le suivi et la supervision du projet - SUIV	4
Le développement - DVLP	5
Les tests - TEST	4
La gestion de configuration - CONF	3
La gestion des sous-traitants - SSTR	4
La gestion de la qualité - QUAL	5
La capitalisation des acquis - CPTL	3
Nombre de processus : 10	**40**

Remarque : Le détail des *pratiques* est disponible dans la version complète du modèle.

3.3. Les facteurs de succès

Les processus sont définis en fonction de l'objectif global de l'entreprise (voir figure 2). Cet objectif est atteint grâce à la mise en œuvre conjointe des pratiques de chaque processus concerné et des facteurs de succès qui les supportent.

Les facteurs de succès peuvent être définis comme des ***éléments d'environnement qui favorisent la mise en place d'un support permettant une exécution optimale des processus***. Leur but n'est donc ni la réalisation d'un délivrable ni la production d'un élément du projet.

Ainsi, une organisation efficace, une bonne gestion, un personnel et une technologie adaptés permettront aux processus de se dérouler dans les meilleures conditions.

L'Organisation

Les processus doivent remplir leur mission de façon efficace de sorte que la société puisse atteindre ses objectifs. La structure organisationnelle en place doit permettre aux processus de s'exécuter dans les meilleures conditions.

Ainsi, cette structure devra entre autre faciliter la circulation de l'information et la communication avec l'environnement (intérieur ou extérieur). Elle devra également permettre la mise à disposition des ressources, et notamment d'un environnement de travail adéquat. Elle devra permettre la définition des objectifs et des responsabilités, ainsi que la prise de décisions importantes, le tout dans l'optique de la réalisation des objectifs de l'entreprise.

La Gestion

La structure en place va permettre une gestion efficace des ressources mises à disposition et un déroulement optimum des processus.

Dans cette optique, la gestion veillera à la définition des procédures à suivre, des processus à exécuter, des normes (de qualité et/ou de quantité) à respecter. Elle comprendra également l'allocation des ressources, la définition et l'implémentation des mécanismes de contrôles à appliquer, et les mécanismes de prise des décisions pour corriger les écarts éventuels.

Les Ressources Humaines

Les ressources humaines constituent assurément la ressource la plus complexe à gérer car la plus instable. Le succès des processus dépend de la disponibilité de toutes les ressources dont ils ont besoin, mais également de l'adéquation des ressources dont ils disposent et du rendement potentiel et effectif de celles-ci.

Les pratiques de suivi nécessitent que les responsabilités aient été attribuées de façon à garantir l'indépendance et l'objectivité de leur résultat.

La Technologie

Comme pour les ressources humaines, les processus ne pourront s'exécuter sans le support d'outils (informatiques ou autres). La performance ou plutôt l'adéquation des outils utilisés aura une influence directe sur le déroulement des processus et la réalisation de leurs objectifs.

Remarque : Le détail des *facteurs des succès* est disponible dans la version complète du modèle.

4. Evaluation des pratiques logicielles selon le modèle *OWPL*

Le modèle *OWPL* est conçu de façon à permettre la mise en évidence rapide des pratiques à améliorer, l'établissement d'un plan d'actions simple visant à améliorer ces pratiques et la prise de mesures simple de la variance induite.

L'évaluation des pratiques logicielles selon le modèle *OWPL* tient compte du contexte particulier des PME wallonnes et des contraintes de départs (simplicité des structures des entreprises wallonnes, petite taille des équipes logicielles, moyens limités). La mise en œuvre de l'évaluation devra donc mobiliser un minimum de ressources tout en maintenant une communication permanente avec tous les intervenants de l'évaluation.

L'évaluation est structurée de façon à mettre en évidence les potentialités d'évolution et d'amélioration de l'unité évaluée et/ou à mesurer l'évolution entre deux évaluations. L'évaluation n'est pas une fin.

4.1. Méthode d'évaluation OWPL

L'évaluation porte sur un ou plusieurs processus sélectionnés sur base des résultats d'une évaluation antérieure (micro-évaluation ou évaluation *OWPL*), sur demande explicite de l'entreprise concernée, ou suite à une analyse de la chaîne des valeurs de cette entreprise.

La maturité des processus est estimée sur base de l'analyse des pratiques qui les composent. Chaque pratique est évaluée par rapport à la qualité de son contenu et à son degré d'institutionnalisation.

Les facteurs de succès qui supportent le déroulement des processus sont évalués uniquement par rapport à leur qualité.

4.2. Evaluation des pratiques

Les pratiques sont évaluées en référence à leur description dans le modèle *OWPL*. L'évaluation devra mettre en évidence si chaque pratique produit bien les sorties qu'elle est supposé produire, la manière dont ces sorties sont produites (niveau de maturité) et si c'est le cas pour certains ou pour tous les projets (degré d'institutionnalisation).

4.3. Evaluation des facteurs de succès

Les facteurs de succès sont évalués pour l'ensemble de l'organisation. Ils sont évalués en référence à leur description dans le modèle *OWPL*. L'évaluation des facteurs de succès devra mettre en évidence la mesure dans laquelle les processus bénéficient du support nécessaire à leur bon déroulement.

4.4. Interprétation des résultats

La méthode de cotation[xi] et d'interprétation des résultats a été définie de façon empirique sur base des premières évaluations réalisées au moyen du modèle.

4.5. Questionnaire d'évaluation

Le questionnaire utilisé dans le cadre de ces évaluations est basé sur le modèle *OWPL* et est disponible par ailleurs[x].

5. Bibliographie

[i] BRODMAN J. G. & JOHNSON D. L.., *What Small Businesses and Small Organisations say about CMM ?* dans Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italie, Mai 1994.

[ii] BROOKS F.P., *No Silver Bullet: Essence and Accidents of Software engineering*, IEEE Computer, Vol. 20, No. 4, Avril 1987, pp. 10-19.

[iii] BROOKS F.P., *The Mythical Man-Month*, Essays on Software Engineering, Anniversary Edition, Addison-Wesley, 1995.

[iv] CAPUTO K., *CMM : Implementation Guide Choreographing Software Process Improvement*, Addison-Wesley, 1998.

[v] DEMING W. E., *Out of the crisis*, MIT, 1997.

[vi] DYMOND K.D., *Le guide du CMM : Introduction au modèle de maturité CMM*, Cepadues, 1997.

[vii] El EMAN K., DROUIN J.-N. & Melo, W., *SPICE : The Theory and Practice of Software Process Improvement and Capability Determination*, IEEE, COMPUTER SOCIETY, 1998.

[viii] FAIRLEY R. E., *Software Engineering Concepts*, McGraw-Hill International, 1985.

[ix] HABRA N., NIYITUGABIRA E. & RENAULT A., *Modèle OWPL - Glossaire*, OWPL-FUNDP 1999

[x] HABRA N., NIYITUGABIRA E. & RENAULT A., *Modèle OWPL - Questionnaire*, OWPL-FUNDP 1999

[xi] HABRA N. & RENAULT A., *Modèle OWPL - Méthode de cotation*, OWPL-FUNDP 2000

[xii] HUMPHREY W. S., *A Discipline for Software Engineering*, Addison-Wesley, 1995.

[xiii] HUMPHREY W. S., *Introduction to the Personal Software Process*, Addison-Wesley, 1997.

[xiv] HUMPHREY W.S., *Managing the Software Process*, SEI Series in Software Engineering, Addison-Wesley, 1991.

[xv] ISO/IEC JTC 1/SC 7, ISO/IEC TR 15504, 1998.

[xvi] JOHNSON D.L.. & BRODMAN J.G., *Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects*, Software Process Newsletter, N° 8, IEEE, Winter 1997.

[xvii] KOCH G., *Process Assessment : the Bootstrap Approach*, Information and Software Technology, Vol30, N°6/7, 1993.

[xviii] LOBET-MARIS Cl., *Utilisation des systèmes d'information interorganisationnels par les PME belges - Rapport final* - CITA-FUNDP - novembre 1997.

[xix] NIYITUGABIRA E., *Adéquation du modèle CMM aux PME et pistes d'adaptation*, OWPL-FUNDP, juillet 1998.

[xx] PAULK M. C., CURTIS B., CHRUSSIS M. B. & WEBER C., *Capability Maturity Model for* Software, Version 1.1", SEI, CMU/SEI-93-TR-24, février 1993.

[xxi] PAULK M. C., WEBER C., GARCIA S. M., CHRUSSIS M. B. & BUSH M., *Key Practices of the Capability Maturity Model, Version 1.1*, SEI, CMU/SEI-93-TR-25, février 1993.

[xxii] PAULK M., et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, 1995.

[xxiii] RENAULT A.., *Etude critique d'une utilisation du modèle SPICE*, OWPL-FUNDP, juillet 1998.

[xxiv] RENAULT A.., *Micro-évaluation des Pratique Logicielles - Etude de cas*, OWPL-FUNDP, janvier 1999.

[xxv] SEI, *Benefits of CMM-Based Software Process Improvement : Initial Results*, CMU/SEI-94-TR-013, ESC-TR-94-013, Août 1994.

[xxvi] SEI, *Process Tailoring and the Software Capability Maturity Model, Technical Report*, CMU/SEI-94-TR-024, ESC-TR-94-024, Novembre 1995.

[xxvii] ZAHRAN S., *Software Process Improvement : Practical Guidelines for Business Successes*, Addison-Wesley, 1997.

[xxviii] ZUBROW D., et. al., *Maturity Questionnaire*, SEI, CMU/SEI-94-SR-007, Avril, 1994.

Document annexe n°2

Le glossaire du modèle OWPL version 1.2

Modèle OWPL

Glossaire

N. Habra, E. Niyitugabira, A. Renault

Institut d'Informatique
FUNDP - Charleroi

OWPL - Glossaire v1.2 .doc, version du 06/08/1999

Table des matières

A

Acquis

Expérience accumulée dans le passé et susceptible d'enrichir les projets en cours ou futurs. [OWPL]

Audit

Revue indépendante d'un ou de plusieurs résultats d'un travail pour évaluer leur conformité aux spécifications, aux normes, aux attentes contractuelles ou à d'autres critères [IEEE-STD-610]

B

Bilan de clôture de projet

Analyse a posteriori d'un projet. Le bilan de clôture sert de base pour la capitalisation des acquis en détaillant l'expérience (positive/négative) vécue lors de ce projet. [OWPL]

C

Cadre d'apprentissage

Ensemble des documents (descriptions des processus, manuel de qualité, etc.) et des ressources permettant un accès permanent à ces documents dans le but de faire bénéficier tous les intervenants du projet de l'expérience des projets antérieurs. [OWPL]

Cahier des charges

Document de travail qui, précisant les besoins de l'utilisateur dans le cadre d'objectifs datés et quantifiés, permet à d'éventuels fournisseurs de proposer une ou plusieurs solutions chiffrées qui seront réalisées suivant les dispositions contractuelles retenues. [F. GUERIN, Maîtriser l'informatique, Delmas, 1990, D2]

Capitalisation des acquis

Ensemble de pratiques de haut niveau permettant une analyse critique des processus en cours ou achevés afin de déceler les bonnes pratiques à généraliser et celles à améliorer (ou à exclure) à l'avenir. [OWPL]

Codage

Traduction de la description technique du système en code opérationnel, compte tenu de l'environnement d'exploitation. [OWPL]

Cohérence

Degré d'uniformité, de normalisation et d'absence de contradiction entre documents, composants ou éléments constitutifs d'un système. [IEEE-STD-610]

Conception

Description technique de la structure et des composants du système permettant la mise en œuvre des fonctionnalités demandées par le client. [OWPL]

Configuration

Ensemble de caractéristiques fonctionnelles et physiques du produit et de son environnement logiciel et matériel. [OWPL]

Contraintes associées aux tâches

Ensemble des contraintes qui conditionnent l'exécution des tâches d'un projet. Ces contraintes peuvent être cumulatives (disponibilité datée de différentes ressources), disjonctives (simultanéité/disjonction entre tâches) ou dites de localisation temporelle (en fonction de leur date de début/fin). [OWPL]

Cycle de vie

Ensemble des étapes nécessaires à la réalisation et l'exploitation (et éventuellement le retrait) d'un produit logiciel. [OWPL]

D

Délivrable

Produit du travail conditionnant le passage d'une phase à une autre du cycle de vie. [OWPL]

Développement

Ensemble de pratiques en relation directe avec la production / l'amélioration d'un produit logiciel. Sont concernées l'analyse fonctionnelle, la conception, le codage, la documentation, les tests, la validation, la mise en exploitation et la maintenance corrective. [OWPL]

Documentation

Rédaction de documents destinés à aider le client à installer, utiliser et gérer de façon optimale le produit logiciel. Par extension, le produit de cette activité. [OWPL]

E

Elément de configuration

Tout produit du travail (produit logiciel, code, compilateur, matériel) ou ensemble de produits du travail identifiés à un moment donné du cycle de vie du logiciel. Un élément de configuration peut être mis en relation avec les autres éléments de configuration d'une même version du produit logiciel et avec les versions antérieures du même élément de configuration. [OWPL]

Entrées

Elément utile ou indispensable à la réalisation d'une pratique. Ces éléments sont généralement produits par d'autres pratiques. [OWPL]

Exigences

Ensemble de propriétés essentielles auxquelles un système doit répondre. [OWPL]

Evaluation de processus

Examen structuré des processus d'une organisation pour déterminer la capacité de ces processus à atteindre leurs objectifs en respectant les contraintes de qualité, de coûts et de délais. Cet examen sera implémenté eu égard à un modèle de référence. [OWPL]

F

Facteur de succès

Eléments d'environnement (organisation, gestion, technologie, ressources humaines) qui favorisent la mise en place d'un support permettant une exécution optimale des processus. [OWPL]

Fonctionnel

En rapport avec les fonctionnalités du produit logiciel. [OWPL]

Formaliser

Exprimer (des exigences, une solution, une procédure, etc.) suivant des formes prédéfinies (règles de référence, formulaire, langage, modèle, etc.). [OWPL]

G

Gestion de la configuration

Discipline s'appuyant sur des directives techniques et administratives en vue d'identifier et de documenter les caractéristiques physiques et fonctionnelles d'un élément de configuration, de contrôler les changements apportés à ces caractéristiques, d'enregistrer et d'indiquer l'état de la mise en œuvre du traitement de ces changements, et de vérifier la conformité aux exigences spécifiées.
[IEEE-STD-610]

I

Intégration

Assemblage de différents composants issus du développement pour produire l'ensemble ou un sous-ensemble prédéfini des fonctionnalités du produit logiciel final. [OWPL]

Institutionnalisation

Mise en place d'une infrastructure et d'une culture d'entreprise appuyant les méthodes, les pratiques et les procédures organisationnelles de façon à ce qu'elles soient appliquées à tous les projets et qu'elles continuent d'être appliquées après le départ de ceux qui les ont définies à l'origine. [OWPL]

Intégrité

Propriété d'un produit dont la stabilité n'est pas menacée par la modification d'un ou de plusieurs de ses composants. [OWPL]

J

Jeu de tests

Description de la combinaison d'un ensemble d'éléments (entrées, sorties, environnement) permettant d'évaluer la conformité d'une ou plusieurs caractéristiques (fiabilité, sécurité, performance, etc.) d'un produit par rapport aux exigences. [OWPL]

L

Logiciel

Voir produit logiciel.

M

Maintenance

Action de modifier un système logiciel ou un de ses composants, après livraison, pour remédier à ses défauts éventuels, améliorer sa performance ou encore l'adapter à un nouvel environnement. [IEEE-STD-610]

Manuel qualité

Document décrivant les dispositions générales prises par l'entreprise pour obtenir et maintenir la qualité de ses produits ou services. [ISO-8402]

Maturité de processus

Mesure de la capacité d'un processus à atteindre ses objectifs d'une façon optimale eu égard à son environnement et aux ressources dont il dispose. [OWPL]

N

Non fonctionnel

Propriété d'un produit ne portant pas sur ses fonctionnalités, mais sur ses contraintes d'utilisation et/ou de réalisation. [OWPL]

Norme de qualité logicielle

Exigence obligatoire utilisée et développée pour établir une démarche uniforme et disciplinée de développement logiciel. [CMM]

O

Objectif de pratique

Description synthétique de la portée, des limites et de l'intention de la pratique. La réalisation effective de la pratique doit concourir à atteindre l'objectif du processus auquel elle appartient. [OWPL]

Objectif de processus

Description de l'objectif fonctionnel global d'un processus. L'objectif du processus est défini en fonction de l'objectif global des processus logiciels. [OWPL]

Objectifs logiciels

Description de l'objectif de l'ensemble des processus logiciels. [OWPL]

P

Plan d'actions

Ensemble d'actions structurées dans le temps qui seront implémentées pour améliorer le niveau de maturité d'une pratique ou d'un processus. [OWPL]

Phase

Subdivision du cycle de vie se terminant par la production d'un délivrable. [OWPL]

Planning

Document(s) reprenant l'ensemble des pratiques à mettre en œuvre, les ressources disponibles et leur agencement dans le temps afin de garantir la bonne fin du projet tout en assurant une consommation optimale de ces ressources. [OWPL]

Poids d'une pratique

Une valeur intrinsèque (subjective) attribuée à la pratique pour souligner son importance dans la réalisation de l'objectif du processus auquel elle appartient. [OWPL]

Pratique

Activité d'ingénierie ou de gestion qui contribue à la création d'un délivrable du processus, ou améliore la capacité du processus. [ISO-15504]

Procédure

Description écrite de la marche à suivre pour l'accomplissement d'une tâche donnée. [IEEE-STD-610]

Processus

Ensemble structuré de pratiques nécessaires à la réalisation d'un objectif commun clairement défini. [OWPL]

Processus logiciels

Ensemble des activités, méthodes et pratiques permettant le développement et la maintenance de logiciels et des produits associés. [CMM]

Produit (du travail)

Elément quelconque créé dans le cadre de la définition, de la maintenance ou de l'utilisation d'un processus, y compris les descriptions, plans, procédures, programmes informatiques et la documentation qui s'y rapporte, qu'il soit prévu ou non de livrer l'élément en question au client ou à l'utilisateur final. [CMM]

Produit logiciel

Totalité ou élément constitutif de l'ensemble complet des programmes informatiques, des procédures, ainsi que la documentation et les données afférentes devant être livrés au client ou à l'utilisateur final. [IEEE-STD-610]

Q

Qualité

Mesure dans laquelle un système, un composant ou un processus satisfait aux exigences énoncées. [IEEE-STD-610]

R

Ressource

Les ressources représentent les éléments techniques, documents, outils ou méthodes qui aident à la réalisation de la pratique. [OWPL]

Revue formelle

Réunion au cours de laquelle un produit du travail est présenté aux personnes concernées pour obtenir leurs commentaires et leur approbation. [OWPL]

Risque

Possibilité d'un écart par rapport aux prévisions ; et par extension, la source même de cet écart. [OWPL]

S

Sortie

Elément concret attendu d'une pratique. Son existence confirme la bonne exécution de la pratique. [OWPL]

T

Tâche

Série d'instructions traitée comme unité élémentaire de travail. [IEEE-STD-610]

Test

Application du (des) jeu(x) de tests à l'entièreté ou à une partie du produit logiciel. [OWPL]

Traçabilité

Mesure dans laquelle il est possible d'établir une relation entre deux ou plusieurs produits issus du processus, en particulier les produits ayant entre eux une relation de succession (prédécesseur / successeur) ou d'asservissement (supérieur/subordonné). [IEEE-STD-610]

V

Validation

Processus d'évaluation du logiciel pendant le processus logiciel ou à la fin de celui-ci pour déterminer s'il répond aux exigences énoncées. [IEEE-STD-610]

Vérification

Processus d'évaluation du logiciel pour déterminer si les produits issus d'une phase donnée du développement répondent aux conditions imposées au début de cette phase. [IEEE-STD-610]

```
function exportTerme(form,xml,xslt)
{
        form.output.value = makeXML();
}


function exportTermeHTML(form,xml,xslt)
{
        var xmlDOC = makeXML();
        xml.async = false;

        // passe une chaîne XML au parser

        xml.loadXML(xmlDOC);
        form.output.value = xml.transformNode(xslt.XMLDocument);

}

function makeXML()
{
        var xmlCode = "";

        var I;
        for (I=0;I < termes.length;I++)
             if (termes[I] != null)
                  xmlCode += termes[i].toXML();

        return element("termeglossaires","",xmlCode);
}

function element(name,attributes,content)
{
  var result = "<" + name;
  if (attributes != "")
     result += " "+attributes;
  result += ">";
  result += content;
  result += "</" + name + ">\r";
  return result;
}



function escapeXML(string)
{

        var result = "",
            I,
           c;
      for (I=0; I < string.length;I++)
      {

        c = string.charAt(i);
        if (c=='<')
             result += "&lt;"
        else if (c=='&')
             result += "&amp;";
        else
             result += c;

      }
      return result;

}


// déclare 2 objets jscript objet terme

function Terme(intit,sign,src)
{
        this.intitule = intit;
      this.signification = sign;
      this.source = src;
      this.toXML = terme_toXML;
}
```

```
function terme_toXML()
{
       var resultintit = element("intitule","",escapeXML(this.intitule)),
          resultsign = element("signification","",escapeXML(this.signification)),
          resultsrc = element("source","",escapeXML(this.source));
     return element("terme","",resultintit+resultsign+resultsrc);
}


function exists(tableau,nomTerme)
{    var result=0;
     var I=0;

     while ((!(result))&&(I<(tableau.length)+ 1))
      {
           result = (nomTerme==tableau[I]);
           I++;
       }
 return result;

}


function deleteTermeList(t,tl)
{

  var trouve=0;
  var i = 0;
  var l = tl.length-1;

  while ((!(trouve))&&(I<(tl.length)+1))
  {

    trouve = (t==tl[i]);
    I++;
  };

  if (trouve)
   {
     i--;
     for(I;I < tl.length; I++)
     {
       tl[I]=tl[I+1];
     };

     tl[l]= null;

   };


}

function EditTerme(form)
{
       var termeList = form.termeList,
       pos = termeList.selectedIndex,
       termeToEdit;

       if(pos!= -1)
       {
         form.intitule.value = termes[termeList.options[pos].value].intitule;
        form.signification.value = termes[termeList.options[pos].value].signification;
        form.source.value = termes[termeList.options[pos].value].source;
        form.BoutonAjouter.value="Enregistrer"

       }

     else alert("veuillez choisir le terme à éditer svp!")

}
```

Document annexe n°4

Signature d'applets Java

Writing Advanced Applications

Chapter 10: Signed Applets

[<<BACK] [CONTENTS] [NEXT>>]

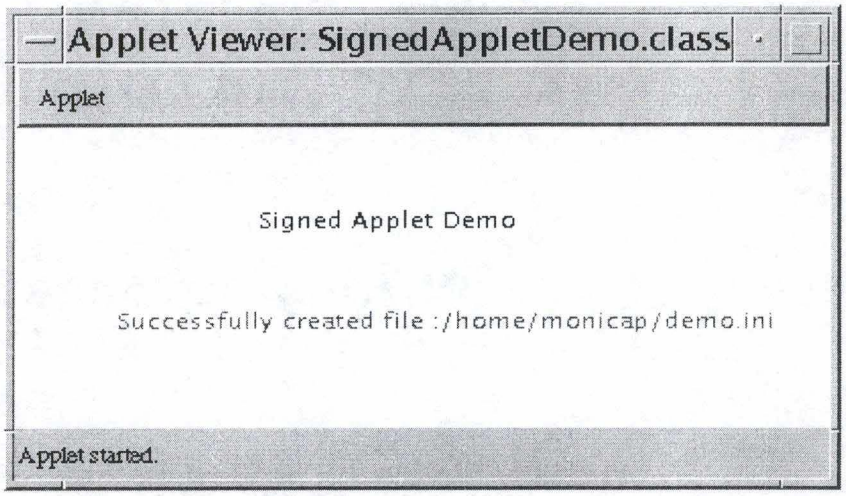
A policy file can be defined to require a signature on all applets or applications that attempt to run with the policy file. The signature is a way to verify that the applet or applicationis from a reliable source and can be trusted to run with the permissions granted in the policy file.

If a policy file requires a signature, an applet or application can get the access granted by the policy file only if it has the correct signature. If the applet or application has the wrong signature or no signature, it will not get access to the file.

This section walks through an example of signing an applet verifying the signature, and running the applet with a policy file.

Signed Applet Example

The policy file granting access can be set up to require or not require a signature. If a signature is required, the applet has to be bundled into a Java ARchive (JAR) file before it can be signed. This example shows you how to sign and grant permission to an applet so it can create `demo.ini` in the user's home directory when it executes in Applet Viewer.



These files are used for the example. You can copy them to or create them in your working directory.

- SignedAppletDemo.java file containing the applet code
- Write.jp policy file granting access to the user's home directory
- Applet tag embedded in the SignedApplet.html file:

```
<applet code="SignedAppletDemo.class"
        archive="SSignedApplet.jar"
        width=400 height=400>
        <param name=file value="/etc/inet/hosts">
</applet>
```

Usually an applet is bundled and signed by an intranet developer and handed off to the end user who verifies the signature and runs the applet. In this example, the intranet developer performs Steps 1 through 5 and Ray, the end user, performs Steps 6 through 8. But, to keep things simple, all steps occur in the same working directory.

1. Compile the applet
2. Create a JAR file
3. Generate Keys
4. Sign the JAR file
5. Export the Public Key Certificate
6. Import the Certificate as a Trusted Certificate
7. Create the policy file
8. Run the applet

Intranet Developer

Susan, the intranet developer, bundles the applet executable in a JAR file, signs the JAR file, and exports the public key certificate.

1: Compile the Applet

In her working directory, Susan uses the `javac` command to compile the `SignedAppletDemo.java` class. The output from the `javac` command is the `SignedAppletDemo.class`.

```
javac SignedAppletDemo.java
```

2: Make a JAR File

Susan then stores the compiled `SignedAppletDemo.class` file into a JAR file. The `-cvf` option to the `jar` command creates a new archive (c), using verbose mode (v), and specifies the archive file name (f). The archive file name is `SignedApplet.jar`.

```
jar cvf SignedApplet.jar SignedAppletDemo.class
```

3: Generate Keys

A JAR file is signed with the private key of the creator of the JAR file and the signature is verified by the recipient of the JAR file with the public key in the pair. The certificate is a statement from the owner of the private key that the public key in the pair has a particular value so the person using the public key can be assured the public key is authentic. Public and private keys must already exist in the keystore database before jarsigner can be used to sign or verify the signature on a JAR file.

Susan creates a `keystore` database named `compstore` that has an entry for a newly generated public and private key pair with the public key in a certificate using the `keytool` command.

In her working directory, Susan creates a keystore database and generates the keys:

```
keytool -genkey -alias signFiles -keystore compstore
        -keypass kpi135 -dname "cn=jones"
        -storepass ab987c
```

This `keytool -genkey` command invocation generates a key pair that is identified by the alias signFiles. Subsequent keytool command invocations use this alias and the key password (`-keypass kpi135`) to access the private key in the generated pair.

The generated key pair is stored in a keystore database called compstore (`-keystore compstore`) in the current directory, and accessed with the compstore password (-

`storepass ab987c`).

The `-dname "cn=jones"` option specifies an X.500 Distinguished Name with a commonName (cn) value. X.500 Distinguished Names identify entities for X.509 certificates. In this example, Susan uses her last name, Jones, for the common name. She could use any common name that suits her purposes.

You can view all keytool options and parameters by typing:

```
keytool -help
```

4: Sign the JAR File

JAR Signer is a command line tool for signing and verifying the signature on JAR files. In her working directory, Susan uses jarsigner to make a signed copy of the `SignedApplet.jar` file.

```
jarsigner -keystore compstore -storepass ab987c
        -keypass kpi135
       -signedjar
       SSignedApplet.jar SignedApplet.jar signFiles
```

The `-storepass ab987c` and `-keystore compstore` options specify the keystore database and password where the private key for signing the JAR file is stored. The `-keypass kpi135` option is the password to the private key, `SSignedApplet.jar` is the name of the signed JAR file, and `signFiles` is the alias to the private key. `jarsigner` extracts the certificate from the keystore whose entry is `signFiles` and attaches it to the generated signature of the signed JAR file.

5: Export the Public Key Certificate

The public key certificate is sent with the JAR file to the enc user who will be using the applet. That person uses the certificate to authenticate the signature on the JAR file. A certificate is sent by exporting it from the `compstore` database.

In her working directory, Susan uses keytool to copy the certificate from `compstore` to a file named `CompanyCer.cer` as follows:

```
keytool -export -keystore compstore -storepass ab987c
       -alias signFiles -file CompanyCer.cer
```

As the last step, Susan posts the JAR and certificate files to a distribution directory on a web page.

End User

Ray, the end user, downloads the JAR file from the distribution directory, imports the certificate, creates a policy file granting the applet access, and runs the applet.

6: Import Certificate as a Trusted Certificate

Ray downloads `SSignedApplet.jar` and `CompanyCer.cer` to his home directory. Ray must now create a keystore database (`raystore`) and import the certificate into it using the alias `company`. Ray uses `keytool` in his home directory to do this:

```
keytool -import -alias company -file
        CompanyCer.cer -keystore
       raystore -storepass abcdefgh
```

7: Create the Policy File

The policy file grants the `SSignedApplet.jar` file signed by the alias `company` permission to create `demo.ini` (and no other file) in the user's home directory.

Ray creates the policy file in his home directory using either `policytool` or an ASCII editor.

```
keystore "/home/ray/raystore";

// A sample policy file that lets a program
// create demo.ini in user's home directory
// Satya N Dodda

grant SignedBy "company" {
  permission java.util.PropertyPermission
    "user.home", "read";
  permission java.io.FilePermission
    "${user.home}/demo.ini", "write";
};
```

8: Run the Applet in Applet Viewer

Applet Viewer connects to the HTML documents and resources specified in the call to `appletviewer`, and displays the applet in its own window. To run the example, Ray copies the signed JAR file and HTML file

to `/home/aURL/public_html` and invokes Applet viewer from his home directory as follows:

```
appletviewer -J-Djava.security.policy=Write.jp
        http://aURL.com/SignedApplet.html
```

> **Note:** Type everything on one line and put a space after `Write.jp`

The `-J-Djava.security.policy=Write.jp` option tells Applet Viewer to run the applet referenced in the `SignedApplet.html` file with the `Write.jp` policy file.

> **Note:** The Policy file can be stored on a server and specified in the `appletviewer` invocation as a URL.

Running an Application with a Policy File

This application invocation restricts `MyProgram` to a sandbox-like environment the same way applets are restricted, but allows access as specified in the `polfile` policy file.

```
java -Djava.security.manager
        -Djava.security.policy=polfile MyProgram
```

Signed Applets in JDK 1.1

JDK 1.1 signed applets can access local system resources if the local system is properly set up to allow it. See the JDK 1.1 Signed Applet Example page for details.

[TOP]

This site is perpetually under construction

Code Signing for Java Applets

<http://www.suitable.com/Doc_CodeSigning.shtml>

by Daniel Griscom, griscom@suitable.com

Previous section: Writing code for Netscape Navigator

Signing code for Netscape Navigator

Summary of Process

Make any necessary changes in the applet's code (see Writing code for Netscape Navigator).

Collect the tools you'll need: a digital ID, and `signtool`. Use `signtool` from an MS-DOS window in Win95 to create a digital signature for the applet's files and pack all the .class files into a .jar file, preserving directories (but stripping off the beginning of each path). Install the results. Enjoy.

> Note: Navigator 3 isn't able to verify signed Java applets, and so can't take advantage of digitally signed applets. Navigator 4 and up can take advantage of digitally signed applets (I've verified this on the Macintosh and under Windows; I expect it is also true on other Navigator platforms).

Collect tools

You'll need two items to do digital signing: a Netscape Object Signing software publishing digital ID, and a DOS program called `signtool.exe`.

> Note: VeriSign is the CA that I used for my certificates. Although it was one of the first, it now has competitors. You can check Netscape's list of CAs which support their products at <https://certs.netscape.com/client.html>, or you can check the Vendors section in the Links page.

To get a digital ID from VeriSign, use Navigator 4.0 or later under Win95 (make sure Java and

JavaScript are enabled). Go to <http://www.verisign.com/>. Click "Developer Tools and Code Signing" and click "Buy Now" next to "Digital ID For Netscape Object Signing". Follow the directions for enrolling for a Class 3 software publishing ID.

> Note: VeriSign used to have a Class 2 digital ID, for use by individual developers. It could be obtained in a manner of minutes, and cost $20/year. They're no longer offering this level of ID; my guess is too many people were buying Class 2 IDs rather than the (much more profitable) Class 3 IDs. Oh, well...

When you've received your ID, it will be automatically installed in a pair of files called "cert7.db" and "key3.db", both in (probably) "C:\Program Files\Netscape\users\<yourName>". This is the digital certificate database used directly by Navigator. You don't need to export anything, but when you use the code signing tools below you'll need to specify this directory, the name of the ID to use within the database, and the password for the database (if any). (Make a backup copy of the two .db files in case things get wiped out.)

> Note: you can also create your own test certificates. For more information, see Creating and Installing Test Certificates.

> Note: you can use `signtool` to create your own test certificate if you like. For instructions on how see Netscape's page *Generating Test Object-Signing Certificates* at <http://developer.netscape.com/docs/manuals/signedobj/signtool/signcert.htm>. Beware, though: this will only work if you have a Navigator password (unlike me): if you don't, you'll get the error message `signtool: failure authenticating to key database: Security I/O error.`

Netscape's `signtool`'s downloading page is at <http://developer.netscape.com/software/tools/index.html?content=/software/signedobj/jarpack.html>. The version for Windows 95 (as of 9/3/99) is at <http://developer.netscape.com/software/signedobj/signtool11/signtool11WIN95.zip>. You can find documentation on `signtool` at the page *Signing Software with Netscape Signing Tool 1.1*, at <http://developer.netscape.com/docs/manuals/signedobj/signtool/index.htm>.

Make sure that `signtool` is placed in one of the directories in your PATH, so that MS-DOS will be able to find it.

Set up a directory for signing

Create a top-level directory for the signing. Within that directory, create a subdirectory containing all the .class files for your applet (I called mine "MyApplet"). Within the subdirectory, place copies of all .class files in their directories. Top level .class files should be right inside this directory, and all package .class files should be in subdirectories with the package names (e.g. all my "util" package .class files are inside the directory "MyApplet\util").

Find Navigator's digital ID database directory

For each browser user, Netscape maintains a directory holding various items, including that user's digital ID database. You'll need to specify this directory when using the code signing tool so that the tool will be able to find the public and private components of your key.

This directory is (usually) "c:\program files\netscape\users\<yourName>". To make sure, search for a directory containing the files "cert7.db" and "key3.db" (which contain your public certificate and private key, respectively). For safety, you might want to copy these files to a secure place.

Find name of your digital ID

Now that you've found the digital ID database, you need to know the exact name of your digital ID. To do this, use `signtool` to list the contents of the database:

```
signtool -d"<DATABASE directory>" -L
```

In my case, I typed this:

```
signtool -d"c:\program files\netscape\users\griscom" -L
```

`signtool` will print out a list of all certificates. Yours will have some long name based on your name, and MUST have a "*" to its left (indicating it is available for signing). I got the following results:

```
using certificate directory: c:\program files\netscape\users\griscom
S Certificates
- ------------
  AT&T Certificate Services
  Thawte Personal Premium CA
  GTE CyberTrust Secure Server CA
  Verisign/RSA Commercial CA
  AT&T Directory Services
  GTIS/PWGSC, Canada Gov. Web CA
  Thawte Personal Freemail CA
  Thawte Server CA
  GTIS/PWGSC, Canada Gov. Secure CA
  MCI Mall CA
  VeriSign Class 4 Primary CA
  United States Postal Service CA
  KEYWITNESS, Canada CA
  Netscape Export Control Policy CA
  BBN Certificate Services CA Root 1
  Thawte Personal Basic CA
  CertiSign BR
  VeriSign Class 3 Primary CA
  Canada Post Corporation CA
  Integrion CA
  IBM World Registry CA
  Uptime Group Plc. Class 1 CA
  VeriSign Class 1 Primary CA
  VeriSign Class 2 Primary CA
  VeriSign, Inc. - VeriSign, Inc.
  Uptime Group Plc. Class 2 CA
  Thawte Premium Server CA
  Uptime Group Plc. Class 3 CA
  Verisign/RSA Secure Server CA
  GTE CyberTrust Root CA
  Uptime Group Plc. Class 4 CA
* Daniel T Griscom's VeriSign Trust Network ID
------------
```

So: my ID name is "Daniel T Griscom's VeriSign Trust Network ID", and can be used for signing (phew). Note the list of the CAs who's CA certificates are installed in my browser, ready to validate digital IDs. If you only want to see the signing certificates, you can use `signtool`'s `-l` option instead of the `-L` option.

Find password for your digital ID

If you have set a Navigator/Communicator password, you'll need this password for access to the database. Being a trusting soul I haven't set one, so the batch file below shows an empty password. If you have set one, you can include it in the batch file (which is insecure), or remove the password argument from the `signtool` line and type in your password each time you sign your applet (which is secure).

> Note: I have had a report that `signtool` 1.1 on WinNT 4.0 with Service Patch 3 fails with a message "PROBLEM signing data (Out of memory)" when a password is included in the command line. When the developer removed the password from the command line, `signtool` put up a dialog prompting for the password and then properly signed the applet. I don't know how often this occurs, but if you see the above message then suspect this bug.

Create a .jar signing batch file

Life's a lot easier when you let the computer do the grunt work. So, here is a DOS batch file that creates a signed .jar archive for all files in a given subdirectory. Create the following DOS batch file called `jarsign.bat`:

```
@ECHO OFF
REM Script to make a directory into a signed .jar file. Takes the directory name
REM its argument; creates a .jar file of the same name in the directory above the
REM specified one. Note: must be run in directory above directory to be signed.

REM I'll set up a couple of variables to make things more readable. You'll need t
REM edit these values to match your setup. If you get an error such as
REM "Out of environment space" then you'll have to increase your environment spac
REM (Boy, do I love DOS.)

REM This is the location of the digital signature database
SET ID_LOC="c:\program files\netscape\users\griscom"

REM This is the name of the digital ID to be used
SET ID_NAME="Daniel T Griscom's VeriSign Trust Network ID"

REM This is the password for the database. I haven't set one for mine,
REM so I don't need anything here (the single space is ignored).
SET ID_PASSWD=" "

REM This is the compression level for the final .jar file. 0 means no
REM compression, 9 means highest compression. Note! it used to be
REM that .jar files had to have no compression to work, but now it seems
REM that it's OK. I don't know when this changed, or with what version
REM of Navigator. signtool's default value is 6. Be warned, and try out
REM whatever you decide.
SET COMPRESSION=9

REM signtool signs the directory and creates the .jar archive.
REM Arguments:
REM    -d[text]    Directory holding digital signature database
REM    -k[text]    Name of ID in digital signature database
REM    -p[text]    Password for the database. NOTE! to be more secure, remove
REM                this argument and you'll be prompted for the password.
REM    -Z[text]    Name of .jar file to be created
REM    -c[digit]   Compression level ("0" - none, "9" - highest).
REM    [rest]      Name of directory to be signed
ECHO *********** About to sign directory using signtool ***********
signtool -d%ID_LOC% -k%ID_NAME% -p%ID_PASSWD% -Z %1.jar -c%COMPRESSION% .\%1

REM Punt the various environment variables
```

```
SET ID_LOC=
SET ID_NAME=
SET ID_PASSWD=
SET COMPRESSION=

ECHO ********** Done creating .jar archive **********
```

Change the ID_LOC, ID_NAME and ID_PASSWD values to correspond to your digital ID directory, name and password, respectively. Make sure that `jarsign.bat` is where MS-DOS can find it (somewhere in your `PATH`). Also, when you execute the batch file you must be in the directory containing the directory to be signed.

> Note: including your password in the text of `jarsign.bat` is a classic security no-no (although it isn't as bad as not having a password at all). If you want to be correct, remove the `-p` argument from the `signtool` line, and you'll be prompted for your database's password each time you run `jarsign.bat`.

Do the actual signing

> Note! Before you run `signcode`, make sure Navigator is shut down!

Change to the directory that contains the directory containing your applet's .class files. Then, run `jarsign` with the name of the applet subdirectory as an argument:

```
jarsign MyApplet
```

You'll see lots of messages scroll up the screen. When done, a new archive with the applet directory's name and the suffix ".jar" will be created.

Verify the signed archive

The first time you create a signed archive you'll want to verify it. Do this by using the -w option for `signtool`:

```
signtool -d"c:\program files\netscape\users\griscom" -w  MyApplet.jar
```

Note: you'll have to change the -d argument to match your own digital ID database directory. You might want to make the following one-line batch file, named `jarcheck.bat`:

```
signtool -d"c:\program files\netscape\users\griscom" -w %1.jar
```

(again, change the directory name), and then use it thusly:

```
jarcheck MyApplet
```

If the archive is signed properly, you'll get a printout of the contents of the signing ID. If not, you won't.

Install the signed archive

Put the signed .jar archive into the web server directory containing the main class of your applet. Change the .html file that invokes the applet so that it mentions the archive:

```
<title>My Wonderful Signed Applet</title>
```

```
<hr>
<applet code="MyApplet.class" ARCHIVE="MyApplet.jar" width=600 height=350>
</applet>
<hr>
```

Possible Problems

If you sign your applet but you still get security exceptions when you run your applet then you code may not be properly using the Netscape Capabilities API to request privileges. Another clue is that you never see Navigator's security dialog, even when your code tries to do secure things. For information on the Capabilities API, see Netscape's document *Java Capabilities API* at <http://developer.netscape.com/library/documentation/signedobj/capsapi.html>, or Joe Bowbeer's article *Signing Applets for Internet Explorer and Netscape Navigator* at <http://ourworld.compuserve.com/homepages/jozart/article/index.html>.

If `signtool` complains "`signtool: PROBLEM signing data (Certificate not approved for this operation)`" then your certificate isn't approved for signing archives. You probably have an Email-signing certificate instead of a software publishing certificate.

A similar error message may indicate that your CA's certificate in your browser isn't marked for certifying software developers. Open the Security Info window, click on "Certificates/Signers" in the left column, select your CA in the list, and then click "Edit". Find the checkbox marked "Accept this Certificate Authority for Certifying software developers" and make sure it is checked.

If `signtool` can't find your private key, then perhaps you haven't been supplying a needed password.

If all else fails, then try this: at each step in the signing process, substitute information that you **know** is wrong. Examples: put in incorrect passwords, change file names, change paths, rename files, etc. If this changes the results (new error message, different error message, etc.) then your original information was probably correct. If not, then either the problem is occurring before that step, or your original information was itself wrong.

Notes

Although it should, Navigator 4.0 doesn't automatically load .gif (and probably .jpg) images from archives. You can, however, write code that will fetch .gif images from your applet's .jar archive. The process is explained in this JavaWorld article: <http://www.javaworld.com/javaworld/jw-07-1998/jw-07-jar.html>.

Files with the suffix `.P12` can be used for moving certificates from machine to machine (in Navigator, choose the "Security Info" menu item, and then click "Certificates/Yours"; there are Import and Export buttons in this screen), but for `signtool` to use the certificates they must be installed into `key3.db` and `cert7.db` files.

If you don't have a digital ID, or you don't want to re-sign your applet again and again while developing, there is hope. Check out the Netscape tech note *Activating Codebase Principals*, at <http://developer.netscape.com/library/technote/security/sectn2.html>. By default, Navigator will let you trust applets with a given digital signature, or from your local hard disk (using `file:` URLs). If you activate codebase principals (meaning that principals, or trusted sources, can depend on where your code is based), Navigator will let you trust applets that come from specific `http:` URLs.

> Sub-note: although the *Activating Codebase Principals* tech note tells you to edit the text file "prefs.js", this isn't always true. On the Macintosh, for instance, you must edit the file "Netscape Preferences", which isn't registered as a text file at all (you must force

a text editor to open it, although once open it's fine). Good luck.

You can also use `signtool` to sign JavaScript scripts that are embedded in .html pages. For more information, see Netscape's documentation at <http://developer.netscape.com/docs/manuals/signedobj/signtool/signscpt.htm>, or Danny Goodman's article at <http://developer.netscape.com/viewsource/goodman_sscripts.html>.

Next section: Writing code for Microsoft Internet Explorer

Disclaimer
Table of Contents

Top of Page | Home | Suitable Systems | Projects | Documents | Demos | Family | Links

---

This site is perpetually under construction

Code Signing for Java Applets

<http://www.suitable.com/Doc_CodeSigning.shtml>

by Daniel Griscom, griscom@suitable.com

Previous section: Writing code for Microsoft Internet Explorer

Signing code for Microsoft Internet Explorer

Summary of Process

Make any necessary changes in the applet's code (see Writing code for Microsoft Internet Explorer).

Collect the tools you'll need: Internet Explorer 4.0 (or later), a Microsoft Authenticode digital ID, `cabarc`, `signcode`, `signer.dll`, `javasign.dll`, and `chktrust`. Use `cabarc` to create a .cab file containing the applet .class and .gif files. Use `signcode` to sign the archive. Use `chktrust` to verify the signed archive. Install the results. Enjoy.

> Note: this procedure has been extensively revamped due to changes in Microsoft's tools. It is now based on the code signing tools included with Microsoft's Java SDK 2.01.
>
> Note: Internet Explorer version 4 was the earliest to recognize signed applets, and only under Windows. The Macintosh version of Explorer doesn't (and may never) recognize signed applets. Ahh, standards...

Collect tools

You'll need seven items to do Explorer digital signing: Microsoft Internet Explorer 4, a Microsoft Authenticode digital ID, a DOS program called `cabarc.exe`, a DOS program called `signcode.exe`, a DOS program called `chktrust`, and two application extensions called `signer.dll` and `javasign.dll`.

Download Internet Explorer 4.0 from the Microsoft web site at <http://www.microsoft.com/windows/ie/default.htm>.

To get a digital ID from VeriSign, use Internet Explorer 4.0 or later under Win95 (make sure Java

and JavaScript are enabled). Go to <http://www.verisign.com/>. Click "Developer Tools and Code Signing" and click "Buy Now" next to "Digital ID For Microsoft Authenticode". Follow the directions for enrolling for a Class 3 software publishing ID.

> Note: VeriSign used to have a Class 2 digital ID, for use by individual developers. It could be obtained in a manner of minutes, and cost $20/year. They're no longer offering this level of ID; my guess is too many people were buying Class 2 IDs rather than the (much more profitable) Class 3 IDs. Oh, well...

When you receive your ID from VeriSign, you will be prompted to store it in a pair of files: your certificate file (with suffix ".spc") and your private key file (with suffix ".pvk"). When you use the code signing tools below you'll need to specify the location of these two files. (Copy them onto a floppy disk for safekeeping.)

> Note: if you get an Authenticode digital ID from Thawte, the ID will be installed directly into the Windows key registry, not into "`.spc`" and "`.pvk`" files. To use such an ID, you have to change the arguments to `signcode` in the `cabsign.bat` file below: instead of using the `-spc` and `-v` arguments, you need to use the `-cn` argument, specifying the certificate name. You can get your certificate name by going to the Content tab under Internet Explorer 4's "View / Internet Options" menu and clicking the "Personal" certificates button. [Thanks for the info, Chris Wooldridge!]

Alternatively, you can create your own test certificates using `makecert` and `cert2spc`. For more information, see Creating and Installing Test Certificates.

Now that you've got your certificate, you'll need to get the other tools. They are all part of Microsoft's Java SDK 2.02, which can be found (9/3/99) at <http://www.microsoft.com/java/download.htm>. (It's a 16MB download to get 300kB of files; sorry.)

The first tool is `cabarc.exe`, a very simple command-line oriented "cab" archiver. There's a much more powerful alternative called `diamond`: it provides WAY more power and complexity than you need for Java code archiving and signing, so I'd suggest you stick with `cabarc`.

> If you're looking to permanently install your Java classes on the user's browser, check out `dubuild.exe`. `dubuild` builds cab files, but uses an OSD manifest rather than a `.inf` file. The advantage is that you can persistently install your java class files on a client machine. `Dubuild` is included in the Microsoft SDK; you can find out more about dubuild at <http://www.microsoft.com/Java/sdk/32/pg/pg_tools_swdist_dubuild.htm>
>
> If you use `dubuild`, the only real difference is the step in the signing batch file that calls `cabarc` will be replaced by a call to `dubuild` (with a different set of parameters, of course).
>
> Note: `Dubuild` generated cab files must be signed; otherwise the browser will ignore the file with no explanation as to what happened. To use the package manager, you also have to use a different set of applet tags, which are documented at <http://www.microsoft.com/Java/sdk/32/pg/pg_pkgdist_dubuild_3.htm>

`signcode.exe` is a very simple command-line oriented code signing tool. `chktrust.exe` is a very simple command-line oriented signature testing tool.

> Note: these tools requires the "Microsoft CryptoAPI" to work. Although this isn't

documented anywhere, this API is installed when you install Internet Explorer 4.0 or later.

Set up a directory for signing

Create a top-level directory for the signing. Within that directory, create a subdirectory containing all the .class files for your applet (I called mine "MyApplet"). Within the subdirectory, place copies of all .class files in their directories. Top level .class files should be right inside this directory, and all package .class files should be in subdirectories with the package names (e.g. all my "util" package .class files are inside a subdirectory called "util"). Include all images files as well (these are utilized by Explorer, although not Navigator).

Create a .cab signing batch file

Create the following DOS batch file called `cabsign.bat`:

```
@ECHO OFF
REM This batch file creates and signs a .cab file. The first argument should be t
REM name of the directory of files to be put into the cabinet (NO terminating "\"
REM please!) The second argument should be the formal name of the
REM applet. The third argument should be low, medium or high (generally low).
REM Note: you should be in the directory containing the directory of
REM files to be CABbbed/signed when you run this.

REM I'll set up a couple of variables to make things more readable. You'll need t
REM edit these values to match your setup. If you get an error such as
REM "Out of environment space" then you'll have to increase your environment spac
REM (Boy, do I love DOS.)

REM This is the location of the digital ID certificate file (.spc). For convenien
REM I put mine in the same directory as my Navigator ID database.
SET CERT_FILE="c:\program files\netscape\users\griscom\mycred.spc"

REM This is the location of the digital ID private key file (.pvk).
SET KEY_FILE="c:\program files\netscape\users\griscom\mycred.pvk"

REM First, create the CAB file. The arguments here are:
REM    -r        Recurse into subdirectories
REM    -p        Preserve path names
REM    -P [arg]  Strip the argument (here "%1\") from the beginning of each path
REM    N  [arg]  Create the given named .cab file
REM    [rest]    Put these files (here "%1\*.*") into the .cab file
REM Note! this does NOT use the -s option to reserve space for the signature;
REM the latest version of signcode (from the Java SDK 2.01) doesn't need this.
ECHO ********** About to create .cab archive using cabarc **********
cabarc -r -p -P %1\ N %1.cab %1\*.*

REM Next, sign the code. Arguments are:
REM    -j javasign.dll    This provides the tools to do Java permission levels
REM    -jp  [arg]         The permission level to be used
REM    -spc [arg]         Software publishing certificate file
REM    -v   [arg]         Private key file
REM    -n   [arg]         Nice name of archive (shown in digital ID dialog)
REM    [arg]              Archive file to be signed (here "%1.cab")
ECHO ********** About to sign archive using signcode **********
signcode -j javasign.dll -jp %3 -spc %CERT_FILE% -v %KEY_FILE% -n %2 %1.cab

REM Finally, timestamp the code. (I put this in a separate command to make each
REM command simpler.) NOTE! for this to work you must have an Internet
REM connection up and running. Arguments are:
REM    -x                 Timestamp the archive; do not sign it (it's already don
```

```
REM     -t    [arg]          The timestamp server's HTTP address (here it's VeriSign
REM     -tr   [arg]          The number of times to try timestamping before giving u
REM     [arg]                Archive file to be timestamped (here "%1.cab")
ECHO *********** About to timestamp .cab archive using signcode ***********
signcode -x -t http://timestamp.verisign.com/scripts/timstamp.dll -tr 5 %1.cab

REM Punt the various environment variables
SET CERT_FILE=
SET KEY_FILE=

ECHO *********** Done timestamping .cab archive ***********
```

Change the CERT_FILE and KEY_FILE values to correspond to the location of your public credential file and private key file, respectively. Make sure that `cabsign.bat` is where MS-DOS can find it (somewhere in your PATH). Also, when you execute the batch file you must be in the directory containing the directory to be signed.

> Note: if you gave your ID a password when you got it from VeriSign, `signcode` will prompt you for the password. There's no way to include a password in `signcode`'s arguments.
>
> Remember: if you use Thawte as your CA you'll have to modify the `signcode` line to use a `-cn` argument instead of the `-spc` and `-v` arguments.

Do the actual signing

Change to the directory that contains the directory containing your applet's .class files. Then, run `cabsign` with the name of the applet subdirectory and the formal name of the applet as an argument.

```
cabsign MyApplet "Super Duper Applet" "low"
```

You'll see lots of messages scroll up the screen. When done, a new file with the applet directory's name and the suffix ".cab" will be created.

> Note: if you don't have an active connection to the Internet when you do this, the timestamping process will fail. The resulting archive will be valid and signed, just not timestamped.
>
> Note: the timestamping URL I give in my batch file above is for VeriSign. If your certificate wasn't issued by VeriSign then you should probably use your CA's timestamping service (although I believe VeriSign's will work, it isn't very fair).

Verify the signed archive

The first time you create a signed archive you'll want to verify it. Do this using `chktrust`:

```
chktrust MyApplet.cab
```

If the archive is signed properly, you'll get a "Security Warning" dialog asking if you want to install and run "Super Duper Applet", which was signed by you (signature verified by your CA). If not, you won't.

Install the signed archive

Put the signed .cab archive into the web server directory containing the main class of your applet.

Change the .html file that invokes the applet so that it mentions the archive:

```
<title>My Wonderful Signed Applet</title>
<hr>
<applet code="MyApplet.class" width=600 height=350>
<param name="CABBASE" value="MyApplet.cab">
</applet>
<hr>
```

> Note: f you need to have an applet with multiple .cab archives, you can use the `CABINETS` applet parameter:
>
> ```
> <param name="CABINETS" value="MyApplet.cab,MyApplet2.cab">
> ```

Possible Problems

Timestamping may fail for a number of reasons:

- `signcode` can't reach the timestamping server because you aren't connected to the Internet
- `signcode` can't reach the timestamping server because there's heavy traffic between you and it
- You may have an incorrect timestamping URL

If you don't see a security dialog when the applet is loaded into Explorer then the applet isn't properly signed and won't be allowed any privileges.

If your signed applet gets a `SecurityExceptionEx[Host]` exception when it tries to do a "dangerous" action, then you'll have to modify your code to assert permission. See Writing code for Microsoft Internet Explorer.

Once in a while I've seen a `.spc` that had **two** certificates instead of one. The first certificate was the CA certificate for the code signing certificate, and the second one was the actual code signing certificate. Problem: `signcode` only looks at the first certificate in a file, so the signing certificate was never seen. I used `certmgr` to see the problem (listing the certificates in the file); once I figured it out I used `certmgr` again to delete the first certificate in the file. (Nasty.)

If all else fails, then try this: at each step in the signing process, substitute information that you **know** is wrong. Examples: put in incorrect passwords, change file names, change paths, rename files, etc. If this changes the results (new error message, different error message, etc.) then your original information was probably correct. If not, then either the problem is occurring before that step, or your original information was itself wrong.

Note on Microsoft Security Levels

Internet Explorer actually supports three different security levels: "high", "medium" and "low." The batch file above assumes that you want the low security level, which grants all possible privileges to the applet. High level means that the applet is restricted to the sandbox, just like an unsigned applet. Medium level sounds more interesting: the applet has access to a certain amount of temporary hard disk space, and can do "user-directed" file I/O.

Problem: for your code to take advantage of the abilities granted at the Medium level you must use Microsoft-proprietary Java classes. This means that if you rely on these capabilities your code will be restricted to running on Microsoft platforms. Since I like conspiracy theories, I view this as one more attempt by Microsoft to lure programmers into writing Java code that only works on Microsoft platforms, thus subverting a major threat to their hegemony. Other, more charitable souls may see

this as Microsoft's generous efforts to enhance Java. You'll have to decide for yourself. For more information, see *Signing a Cabinet File with Java Permissions* at <http://www.microsoft.com/Java/sdk/32/pg/pg_pkgdist_signcode_4.htm>.

Next section: Signing code for both Navigator and Explorer

Disclaimer
Table of Contents

Top of Page | Home | Suitable Systems | Projects | Documents | Demos | Family | Links

---

Document annexe n°5

Exemple des fichiers XML utilisés dans OWPL Manager

Exemple de Contenu du fichier processus.XML qui représente la table PROCESSUS définie dans le schéma relationnel (figure 9)

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='yes'?>


<processuss>

<processus>
      <id>EXIG </id>
      <intitule> </intitule>
      <nom>Gestion des exigences </nom>
      <objectif>La gestion des exigences a pour but de définir de manière
      non équivoque les exigences du client, d'en assurer une compréhension
      commune entre les intervenants et de garantir que leur évolution est
      bien prise en compte dans le cahier des charges.
      </objectif>
      <description>La gestion des exigences comprend la production et la
      maintenance du cahier des charges sur la base des exigences du client
      et de leur évolution. le cahier des charges constituera ensuite la
      base pour l'estimation, la planification, la mise en oeuvre et le
      suivi des activités tout au long du projet. La gestion des exigences
      est l'un des principaux paramètres de stabilisation des processus et
      de reproductibilité du succès.
      </description>
      <poids> </poids>
</processus>

<processus>
      <id>DOCS </id>
      <intitule> </intitule>
      <nom>Documentation </nom>
      <objectif>La gestion des exigences a pour but de définir de manière
      non équivoque les exigences du client, d'en assurer une compréhension
      commune entre les intervenants et de garantir que leur évolution est
      bien prise en compte dans le cahier des charges.
      </objectif>
      <description>La gestion des exigences comprend la production et la
      maintenance du cahier des charges sur base des exigences du client et
      de leur évolution. Le cahier des charges constituera ensuite la base
      pour estimation, la planification, la mise...
      </description>
      <poids> </poids>
</processus>

<processus>
<id>DVLP </id>
<intitule> </intitule>
<nom>Développement </nom>
<objectif>Le processus de développement englobe toutes les étapes liées à
la production du produit logiciel... </objectif>
<description>Le processus de développement intègre et exécute les pratiques
de développement de façon cohérente pour produire le logiciel...
</description>
<poids> </poids>
</processus>

</processuss>
```

Exemple de Contenu du fichier `pratique.XML` qui représente la table PRATIQUE définie dans le schéma relationnel (figure 9)

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='yes'?>

<pratiques>

<pratique>
<id>EXIG/PR01/03 </id>
<intitule> </intitule>
<numero>PR01/03 </numero>
<nom>Analyse des exigences </nom>
<objectif>Inventorier, clarifier et définir de façon non ambiguë les
exigences du client afin d'en assurer une compréhension commune par tous
les intervenants du projet. </objectif>
<description> </description>
<poids>12 </poids>
<processus>EXIG </processus>
</pratique>

<pratique>
<id>EXIG/PR02/03 </id>
<intitule> </intitule>
<numero>PR02/03 </numero>
<nom>Suivi des exigences </nom>
<objectif>Maintenir la cohérence du cahier des charges avec les exigences
tout au long du projet. En cours de projet le client peut imposer de
nouvelles exigences. De même, certaines exigences de départ peuvent devenir
obsolètes. Le prestataire peut également influencer le client dans le choix
d'un environnement, d'un outil, d'une méthode, etc. </objectif>
<description> </description>
<poids>100 </poids>
<processus>EXIG </processus>
</pratique>

<pratique>
<id>DOCS/PR01/02 </id>
<intitule> </intitule>
<numero>PR01/03 </numero>
<nom>Identification des exigences en terme de documentation </nom>
<objectif>Inventorier, clarifier et définir de façon non ambiguë les
documents à produire et leur contenu. </objectif>
<description> </description>
<poids>25 </poids>
<processus>DOCS </processus>
</pratique>

<pratique>
<id>EXIG/PR03/03 </id>
<intitule> </intitule>
<numero>PR02/03 </numero>
<nom>Validation </nom>
<objectif>Evaluer le produit logiciel pour déterminer s'il correspond aux
exigences du client. </objectif>
<description> </description>
<poids>35 </poids>
<processus>EXIG </processus>
</pratique>

</pratiques>
```

Exemple de Contenu du fichier `delivrable.XML` qui représente la table PRATIQUE définie dans le schéma relationnel (figure 9)

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='yes'?>

<delivrables>

<delivrable>
<id-delivrable>DE3850 </id-delivrable>
<nom>Architecture du système </nom>
<intitule> </intitule>
<description>XXXX </description>
</delivrable>

<delivrable>
<id-delivrable>DE0001 </id-delivrable>
<nom>Avenant du cahier des charges </nom>
<intitule> </intitule>
<description> </description>
</delivrable>

<delivrable>
<id-delivrable>DE0002 </id-delivrable>
<nom>Cahier des charges </nom>
<intitule> </intitule>
<description> </description>
</delivrable>

<delivrable>
<id-delivrable>DE0003 </id-delivrable>
<nom>Descriptif des documents à produire </nom>
<intitule> </intitule>
<description> </description>
</delivrable>

<delivrable>
<id-delivrable>DE3246 </id-delivrable>
<nom>Description technique du système </nom>
<intitule> </intitule>
<description>XXXXXXXXX </description>
</delivrable>

<delivrable>
<id-delivrable>DE0004 </id-delivrable>
<nom>Documents types </nom>
<intitule> </intitule>
<description> </description>
</delivrable>

<delivrable>
<id-delivrable>DE0005 </id-delivrable>
<nom>Estimation des ressources </nom>
<intitule> </intitule>
<description> </description>
</delivrable>

</delivrables>
```

Document annexe n°6

API des classes Java développées pour OWPL Manager

All Classes

Accueil
EditionDR
EditionNouveauDR
EditionPratique
EditionProcessus
EditionTerme
JTable2
OM
Primitives
Processus
TermeGlossaire

Class Accueil

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--Accueil
```

public class **Accueil**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener

Fichier : Accueil.java

Classe : Fenêtre Accueille qui explique le projet OWPL Cette fenêtre fait partie de l'application OWPL Manager

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Fields inherited from class javax.swing.JInternalFrame
closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION,

WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

Accueil()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)

Methods inherited from class javax.swing.JInternalFrame

addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable, isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected, setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI,

unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Accueil

```
public Accueil()
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

Class EditionDR

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--EditionDR
```

---

public class **EditionDR**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener

Fichier : EditionDR.java

Classe : Fenêtre EditionDR qui manipule les délivrables et et les ressources du modèle OWPL. Cette fenêtre fait partie de l'application OWPL Manager. Elle est appelée quand on veut ajouter une ressource ou un délivrable à une pratique

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
`javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon`

Inner classes inherited from class javax.swing.JComponent
`javax.swing.JComponent.AccessibleJComponent`

Field Summary

`static javax.swing.JButton`	`Ajouter` Bouton pour ajouter un délivrable ou une ressource à une pratique
`static javax.swing.JButton`	`Annuler` Bouton pour Fermer la fenêtre
`static javax.swing.JButton`	`Editer`

	Bouton Editer pour Editer une ressources ou un délivrable
static java.util.Vector	LignesDonnees Vecteur utilisé pour créer la liste des données
static javax.swing.JList	ListeDonnees Pour afficher la liste des délivrables ou des ressources
static java.util.Vector	ListeDR Vecteur qui contient les données lues dans la table delivrable.xml ou ressource.xml
static java.lang.String	Nature Ce String permet de déterminer la nature d'édition (entrée, sortie ou délivrable)
static javax.swing.JButton	Nouveau Bouton pour ajouter un nouveau délivrable ou une nouvelle ressource
static int	PositionAjout Position où on va ajouter sur la liste des entrées, des sorties ou des ressources d'une pratique
static javax.swing.JButton	Supprimer Bouton pour supprimer une ressource ou un délivrable
static javax.swing.JButton	Visualiser Bouton Visualiser pour Visualiser une ressources ou un délivrable

Fields inherited from class javax.swing.JInternalFrame
closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

EditionDR(java.lang.String Filename)

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)

Methods inherited from class javax.swing.JInternalFrame

addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable, isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected, setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics,

getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

ListeDonnees

public static javax.swing.JList **ListeDonnees**

Pour afficher la liste des délivrables ou des ressources

LignesDonnees

public static java.util.Vector **LignesDonnees**

Vecteur utilisé pour créer la liste des données

ListeDR

public static java.util.Vector **ListeDR**

Vecteur qui contient les données lues dans la table delivrable.xml ou ressource.xml

---

Editer

public static javax.swing.JButton **Editer**

Bouton Editer pour Editer une ressources ou un délivrable

Visualiser

```
public static javax.swing.JButton Visualiser
```

Bouton Visualiser pour Visualiser une ressources ou un délivrable

---

Ajouter

```
public static javax.swing.JButton Ajouter
```

Bouton pour ajouter un délivrable ou une ressource à une pratique

---

Nouveau

```
public static javax.swing.JButton Nouveau
```

Bouton pour ajouter un nouveau délivrable ou une nouvelle ressource

---

Supprimer

```
public static javax.swing.JButton Supprimer
```

Bouton pour supprimer une ressource ou un délivrable

Annuler

```
public static javax.swing.JButton Annuler
```

Bouton pour Fermer la fenêtre

Nature

```
public static java.lang.String Nature
```

Ce String permet de déterminer la nature d'édition (entrée, sortie ou délivrable)

PositionAjout

```
public static int PositionAjout
```

Position où on va ajouter sur la liste des entrées, des sorties ou des ressources d'une pratique

Constructor Detail

EditionDR

```
public EditionDR(java.lang.String Filename)
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

Class EditionNouveauDR

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--EditionNouveauDR
```

---

public class **EditionNouveauDR**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener, javax.swing.event.DocumentListener

Fichier : EditionNouveauDR.java

Classe : Fenêtre EditionNouveauDR qui édite les délivrables et les ressources du modèle OWPL. Cette fenêtre fait partie de l'application OWPL Manager. Elle est appelée quand on veut ajouter ou éditer un délivrable u une ressource à une pratique.

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Field Summary

javax.swing.JButton	Ajouter
javax.swing.JButton	Annuler
static java.util.Vector	Liste

static int	Position
static java.lang.String	SavedDescription
static java.lang.String	SavedNomDR
static java.lang.String	SavedReference
javax.swing.JTextArea	TDescriptionDR
javax.swing.JTextField	TNomDR

Fields inherited from class javax.swing.JInternalFrame
closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

EditionNouveauDR(java.util.Vector ListeDonnee)

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	changedUpdate(javax.swing.event.DocumentEvent de)
void	insertUpdate(javax.swing.event.DocumentEvent de)
void	removeUpdate(javax.swing.event.DocumentEvent de)

Methods inherited from class javax.swing.JInternalFrame

addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable, isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected, setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit,

getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

Ajouter

public javax.swing.JButton **Ajouter**

Annuler

public javax.swing.JButton **Annuler**

TNomDR

public javax.swing.JTextField **TNomDR**

TDescriptionDR

public javax.swing.JTextArea **TDescriptionDR**

Position

public static int **Position**

SavedReference

public static java.lang.String **SavedReference**

SavedNomDR

```
public static java.lang.String SavedNomDR
```

SavedDescription

```
public static java.lang.String SavedDescription
```

Liste

```
public static java.util.Vector Liste
```

Constructor Detail

EditionNouveauDR

```
public EditionNouveauDR(java.util.Vector ListeDonnee)
```

Method Detail

insertUpdate

```
public void insertUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
insertUpdate in interface javax.swing.event.DocumentListener

---

removeUpdate

```
public void removeUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
removeUpdate in interface javax.swing.event.DocumentListener

---

changedUpdate

```
public void changedUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
changedUpdate in interface javax.swing.event.DocumentListener

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

Class EditionPratique

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--EditionPratique
```

---

public class **EditionPratique**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener, javax.swing.event.DocumentListener

Fichier : EditionPratique.java

Classe : Fenêtre EditionPratique qui édite les processus du modèle OWPL. Cette fenêtre fait partie de l'application OWPL Manager. Elle est appelée quand on veut ajouter ou éditer une pratique du modèle à partir de la fenêtre d'édition d'un processus.

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Field Summary

static javax.swing.JButton	Ajouter
javax.swing.JButton	AjouterEntree
javax.swing.JButton	AjouterRessource

javax.swing.JButton	AjouterSortie
javax.swing.JButton	Annuler
static java.util.Vector	IndexEntrees Liste des index des entrées de la pratique sur le vecteur des délivrables
static java.util.Vector	IndexRessources Liste des index des ressources de la pratique sur le vecteur des ressources
static java.util.Vector	IndexSorties Liste des index des sorties de la pratique sur le vecteur des délivrables
static java.util.Vector	LignesE Liste des noms des entrées de la pratique
static java.util.Vector	LignesR Liste des noms des ressources de la pratique
static java.util.Vector	LignesS Liste des noms des sorties de la pratique
static java.util.Vector	ListeEntrees Liste des entrées de la pratique (enregistrements)
static java.util.Vector	ListeRessources Liste des ressources de la pratique (enregistrements)
static java.util.Vector	ListeSorties Liste des sorties de la pratique (enregistrements)
static int	Position Position d'Ajout de la pratique sur la liste des pratique
static int	PositionAjout
static javax.swing.JButton	SupprimerEntree
static javax.swing.JButton	SupprimerRessource
static javax.swing.JButton	SupprimerSortie
static JTable2	TableEntrees table des entrées de la pratique
static JTable2	TableRessources table des ressources de la pratique
static JTable2	TableSorties table des sorties de la pratique
static javax.swing.JTextField	TNom

static javax.swing.JTextArea	TObjectif
static javax.swing.JTextField	TPoids
static javax.swing.JTextField	TPratique
static javax.swing.JTextField	TProcessus
static javax.swing.JTextField	TReference

Fields inherited from class javax.swing.JInternalFrame
closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

EditionPratique()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	changedUpdate(javax.swing.event.DocumentEvent de)
void	insertUpdate(javax.swing.event.DocumentEvent de)
void	removeUpdate(javax.swing.event.DocumentEvent de)

Methods inherited from class javax.swing.JInternalFrame
addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable,

isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected, setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent,

processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

Ajouter

public static javax.swing.JButton **Ajouter**

Annuler

public javax.swing.JButton **Annuler**

AjouterEntree

public javax.swing.JButton **AjouterEntree**

SupprimerEntree

public static javax.swing.JButton **SupprimerEntree**

AjouterSortie

public javax.swing.JButton **AjouterSortie**

SupprimerSortie

public static javax.swing.JButton **SupprimerSortie**

AjouterRessource

```
public javax.swing.JButton AjouterRessource
```

SupprimerRessource

```
public static javax.swing.JButton SupprimerRessource
```

TNom

```
public static javax.swing.JTextField TNom
```

TProcessus

```
public static javax.swing.JTextField TProcessus
```

TPratique

```
public static javax.swing.JTextField TPratique
```

TReference

```
public static javax.swing.JTextField TReference
```

TObjectif

```
public static javax.swing.JTextArea TObjectif
```

TPoids

```
public static javax.swing.JTextField TPoids
```

Position

```
public static int Position
```

Position d'Ajout de la pratique sur la liste des pratique

LignesE

```
public static java.util.Vector LignesE
```

Liste des noms des entrées de la pratique

LignesS

```
public static java.util.Vector LignesS
```

Liste des noms des sorties de la pratique

LignesR

```
public static java.util.Vector LignesR
```

Liste des noms des ressources de la pratique

ListeEntrees

```
public static java.util.Vector ListeEntrees
```

Liste des entrées de la pratique (enregistrements)

ListeSorties

```
public static java.util.Vector ListeSorties
```

Liste des sorties de la pratique (enregistrements)

ListeRessources

```
public static java.util.Vector ListeRessources
```

Liste des ressources de la pratique (enregistrements)

IndexEntrees

```
public static java.util.Vector IndexEntrees
```

Liste des index des entrées de la pratique sur le vecteur des délivrables

IndexSorties

```
public static java.util.Vector IndexSorties
```

Liste des index des sorties de la pratique sur le vecteur des délivrables

---

IndexRessources

```
public static java.util.Vector IndexRessources
```

Liste des index des ressources de la pratique sur le vecteur des ressources

---

TableEntrees

```
public static JTable2 TableEntrees
```

table des entrées de la pratique

TableSorties

```
public static JTable2 TableSorties
```

table des sorties de la pratique

TableRessources

```
public static JTable2 TableRessources
```

table des ressources de la pratique

PositionAjout

```
public static int PositionAjout
```

Constructor Detail

EditionPratique

```
public EditionPratique()
```

Method Detail

insertUpdate

```
public void insertUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
insertUpdate in interface javax.swing.event.DocumentListener

---

removeUpdate

```
public void removeUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
removeUpdate in interface javax.swing.event.DocumentListener

---

changedUpdate

```
public void changedUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
changedUpdate in interface javax.swing.event.DocumentListener

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

Class EditionProcessus

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--EditionProcessus
```

---

public class **EditionProcessus**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener, javax.swing.event.DocumentListener

Fichier : EditionProcessus.java

Classe : Fenêtre EditionProcessus qui édite les processus du modèle OWPL. Cette fenêtre fait partie de l'application OWPL Manager. Elle est appelée quand on veut ajouter ou éditer un processus du modèle.

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Field Summary

static javax.swing.JButton	Ajouter
javax.swing.JButton	AjouterPratique
static javax.swing.JButton	Annuler

javax.swing.JButton	EditerPratique
static java.util.Vector	Entrees vecteur des vecteurs contenant toutes les relations Pratique*Délivrable en entrée
static java.util.Vector	IndexPratiques vecteur des indexes des pratiques du processus édité sur la liste des pratiques
static java.util.Vector	LignesP vecteurs contenant les lignes des pratiques à afficher
static boolean	ModificationPratiques Vrai si la pratique éditée a été modifiée
static javax.swing.JPanel	p3
static javax.swing.JPanel	p3North
static javax.swing.JPanel	PanelBouttonsPratique
static int	Position Position d'ajout du processus sur la liste des processus
static int	PositionAffichage Position relative d'affichage d'une pratique du processus
static int	PositionAjout
static int	PositionReelle Position réelle d'une pratique du processus sur la liste des pratiques
static java.util.Vector	Ressources vecteurs des vecteurs contenant toutes les relations Pratique*Ressource utilisée
static java.lang.String	SavedDescription Référence sauvegardée du processus
static java.lang.String	SavedNom Nom sauvegardé du processus
static java.lang.String	SavedObjectif Objectif sauvegardé du processus
static java.lang.String	SavedReference Référence sauvegardée du processus
static java.util.Vector	Sorties vecteur des vecteurs contenant toutes les relations Pratique*Délivrable en sortie
static javax.swing.JButton	SupprimerPratique
static JTable2	

	TablePratiques table des pratiques du processus
static javax.swing.JTextArea	TDescription
static javax.swing.JTextField	TNom
static javax.swing.JTextArea	TObjectif
static javax.swing.JTextField	TReference
javax.swing.JButton	VisualiserPratique

Fields inherited from class javax.swing.JInternalFrame
closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

EditionProcessus()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	changedUpdate(javax.swing.event.DocumentEvent de)
static void	Inserer(java.util.Vector Liste, java.util.Vector Donnee, java.lang.String Nature)
void	insertUpdate(javax.swing.event.DocumentEvent de)
static void	main(java.lang.String[] args)
void	removeUpdate(javax.swing.event.DocumentEvent de)

Methods inherited from class javax.swing.JInternalFrame

addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable, isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected, setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents,

contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

Ajouter

public static javax.swing.JButton **Ajouter**

Annuler

public static javax.swing.JButton **Annuler**

AjouterPratique

public javax.swing.JButton **AjouterPratique**

EditerPratique

public javax.swing.JButton **EditerPratique**

VisualiserPratique

public javax.swing.JButton **VisualiserPratique**

SupprimerPratique

```
public static javax.swing.JButton SupprimerPratique
```

TNom

```
public static javax.swing.JTextField TNom
```

TReference

```
public static javax.swing.JTextField TReference
```

TObjectif

```
public static javax.swing.JTextArea TObjectif
```

TDescription

```
public static javax.swing.JTextArea TDescription
```

PanelBouttonsPratique

```
public static javax.swing.JPanel PanelBouttonsPratique
```

---

p3North

```
public static javax.swing.JPanel p3North
```

---

p3

```
public static javax.swing.JPanel p3
```

Position

```
public static int Position
```

Position d'ajout du processus sur la liste des processus

SavedReference

```
public static java.lang.String SavedReference
```

Référence sauvegardée du processus

SavedNom

```
public static java.lang.String SavedNom
```

Nom sauvegardé du processus

SavedObjectif

```
public static java.lang.String SavedObjectif
```

Objectif sauvegardé du processus

SavedDescription

```
public static java.lang.String SavedDescription
```

Référence sauvegardée du processus

LignesP

```
public static java.util.Vector LignesP
```

vecteurs contenant les lignes des pratiques à afficher

Entrees

```
public static java.util.Vector Entrees
```

vecteur des vecteurs contenant toutes les relations Pratique*Délivrable en entrée

---

Sorties

```
public static java.util.Vector Sorties
```

vecteur des vecteurs contenant toutes les relations Pratique*Délivrable en sortie

---

Ressources

```
public static java.util.Vector Ressources
```

vecteurs des vecteurs contenant toutes les relations Pratique*Ressource utilisée

---

TablePratiques

```
public static JTable2 TablePratiques
```

table des pratiques du processus

IndexPratiques

```
public static java.util.Vector IndexPratiques
```

vecteur des indexes des pratiques du processus édité sur la liste des pratiques

---

PositionAffichage

```
public static int PositionAffichage
```

Position relative d'affichage d'une pratique du processus

PositionReelle

```
public static int PositionReelle
```

Position réelle d'une pratique du processus sur la liste des pratiques

---

PositionAjout

```
public static int PositionAjout
```

ModificationPratiques

```
public static boolean ModificationPratiques
```

Vrai si la pratique éditée a été modifiée

Constructor Detail

EditionProcessus

```
public EditionProcessus()
```

Method Detail

insertUpdate

```
public void insertUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
insertUpdate in interface javax.swing.event.DocumentListener

---

removeUpdate

```
public void removeUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
removeUpdate in interface javax.swing.event.DocumentListener

---

changedUpdate

```
public void changedUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
changedUpdate in interface javax.swing.event.DocumentListener

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

---

main

```
public static void main(java.lang.String[] args)
```

Inserer

```
public static void Inserer(java.util.Vector Liste,
                           java.util.Vector Donnee,
                           java.lang.String Nature)
```

Class EditionTerme

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--EditionTerme
```

---

public class **EditionTerme**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener, javax.swing.event.DocumentListener

Fichier : EditionTerme.java

Classe : Fenêtre EditionTerme qui édite les terme du modèle OWPL. Cette fenêtre fait partie de l'application OWPL Manager. Elle est appelée quand on veut ajouter ou éditer un terme dans le Glossaire.

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Field Summary

javax.swing.JButton	Ajouter
javax.swing.JButton	Annuler
static int	Position

static java.lang.String	SavedIntitule
static java.lang.String	SavedSignification
static java.lang.String	SavedSource
javax.swing.JTextField	Tidtermeglossaire
javax.swing.JTextField	Tintitule
javax.swing.JTextField	TSignification
javax.swing.JTextField	TSource

Fields inherited from class javax.swing.JInternalFrame
closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

EditionTerme()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	changedUpdate(javax.swing.event.DocumentEvent de)
void	insertUpdate(javax.swing.event.DocumentEvent de)
void	removeUpdate(javax.swing.event.DocumentEvent de)

Methods inherited from class javax.swing.JInternalFrame
addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable, isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected, setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent
addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel,

getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

Ajouter

public javax.swing.JButton **Ajouter**

Annuler

public javax.swing.JButton **Annuler**

Tidtermeglossaire

public javax.swing.JTextField **Tidtermeglossaire**

Tintitule

public javax.swing.JTextField **Tintitule**

TSignification

public javax.swing.JTextField **TSignification**

TSource

```
public javax.swing.JTextField TSource
```

Position

```
public static int Position
```

SavedIntitule

```
public static java.lang.String SavedIntitule
```

SavedSignification

```
public static java.lang.String SavedSignification
```

SavedSource

```
public static java.lang.String SavedSource
```

Constructor Detail

EditionTerme

```
public EditionTerme()
```

Method Detail

insertUpdate

```
public void insertUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
insertUpdate in interface javax.swing.event.DocumentListener

---

removeUpdate

```
public void removeUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
removeUpdate in interface javax.swing.event.DocumentListener

---

changedUpdate

```
public void changedUpdate(javax.swing.event.DocumentEvent de)
```

Specified by:
changedUpdate in interface javax.swing.event.DocumentListener

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

Class JTable2

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JTable
                          |
                          +--JTable2
```

public class **JTable2**
extends javax.swing.JTable

See Also:
Serialized Form

Inner classes inherited from class javax.swing.JTable
`javax.swing.JTable.AccessibleJTable`

Inner classes inherited from class javax.swing.JComponent
`javax.swing.JComponent.AccessibleJComponent`

Fields inherited from class javax.swing.JTable
AUTO_RESIZE_ALL_COLUMNS, AUTO_RESIZE_LAST_COLUMN, AUTO_RESIZE_NEXT_COLUMN, AUTO_RESIZE_OFF, AUTO_RESIZE_SUBSEQUENT_COLUMNS, autoCreateColumnsFromModel, autoResizeMode, cellEditor, cellSelectionEnabled, columnModel, dataModel, defaultEditorsByColumnClass, defaultRenderersByColumnClass, editingColumn, editingRow, editorComp, gridColor, preferredViewportSize, rowHeight, rowMargin, rowSelectionAllowed, selectionBackground, selectionForeground, selectionModel, showHorizontalLines, showVerticalLines, tableHeader

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

JTable2(java.util.Vector Rows, java.util.Vector Columns)

Method Summary

boolean	isCellEditable(int row, int column)

Methods inherited from class javax.swing.JTable

addColumn, addColumnSelectionInterval, addNotify, addRowSelectionInterval, clearSelection, columnAdded, columnAtPoint, columnMarginChanged, columnMoved, columnRemoved, columnSelectionChanged, configureEnclosingScrollPane, convertColumnIndexToModel, convertColumnIndexToView, createDefaultColumnModel, createDefaultColumnsFromModel, createDefaultDataModel, createDefaultEditors, createDefaultRenderers, createDefaultSelectionModel, createDefaultTableHeader, createScrollPaneForTable, editCellAt, editCellAt, editingCanceled, editingStopped, getAccessibleContext, getAutoCreateColumnsFromModel, getAutoResizeMode, getCellEditor, getCellEditor, getCellRect, getCellRenderer, getCellSelectionEnabled, getColumn, getColumnClass, getColumnCount, getColumnModel, getColumnName, getColumnSelectionAllowed, getDefaultEditor, getDefaultRenderer, getEditingColumn, getEditingRow, getEditorComponent, getGridColor, getIntercellSpacing, getModel, getPreferredScrollableViewportSize, getRowCount, getRowHeight, getRowMargin, getRowSelectionAllowed, getScrollableBlockIncrement, getScrollableTracksViewportHeight, getScrollableTracksViewportWidth, getScrollableUnitIncrement, getSelectedColumn, getSelectedColumnCount, getSelectedColumns, getSelectedRow, getSelectedRowCount, getSelectedRows, getSelectionBackground, getSelectionForeground, getSelectionModel, getShowHorizontalLines, getShowVerticalLines, getTableHeader, getToolTipText, getUI, getUIClassID, getValueAt, initializeLocalVars, isCellSelected, isColumnSelected, isEditing, isManagingFocus, isRowSelected, moveColumn, paramString, prepareEditor, prepareRenderer, removeColumn, removeColumnSelectionInterval, removeEditor, removeRowSelectionInterval, reshape, resizeAndRepaint, rowAtPoint, selectAll, setAutoCreateColumnsFromModel, setAutoResizeMode, setCellEditor, setCellSelectionEnabled, setColumnModel, setColumnSelectionAllowed, setColumnSelectionInterval, setDefaultEditor, setDefaultRenderer, setEditingColumn, setEditingRow, setGridColor, setIntercellSpacing, setModel, setPreferredScrollableViewportSize, setRowHeight, setRowMargin, setRowSelectionAllowed, setRowSelectionInterval, setSelectionBackground, setSelectionForeground, setSelectionMode, setSelectionModel, setShowGrid, setShowHorizontalLines, setShowVerticalLines, setTableHeader, setUI, setValueAt, sizeColumnsToFit, sizeColumnsToFit, tableChanged, updateUI, valueChanged

Methods inherited from class javax.swing.JComponent

addAncestorListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getTopLevelAncestor,

getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setForeground, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setLayout, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

JTable2

public **JTable2**(java.util.Vector Rows,

```
java.util.Vector Columns)
```

Method Detail

isCellEditable

```
public boolean isCellEditable(int row,
                              int column)
```

Overrides:

isCellEditable in class javax.swing.JTable

Class OM

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Frame
                          |
                          +--javax.swing.JFrame
                                |
                                +--OM
```

public class **OM**
extends javax.swing.JFrame

See Also:
Serialized Form

Inner classes inherited from class javax.swing.JFrame
`javax.swing.JFrame.AccessibleJFrame`

Field Summary

`static javax.swing.JDesktopPane`	`desktop`
`static java.util.Vector`	`Entrees` Vecteur des Vecteurs Entrées (enregistrements du fichier entree.xml)
`static java.util.Vector`	`ListeDelivrables` Vecteur des Vecteurs Delivrables (enregistrements du fichier delivrable.xml)
`static java.util.Vector`	`ListeNomProcessus` Vecteur qui contient les noms des processus pour faire le choix
`static java.util.Vector`	`ListePratiques` Vecteur des Vecteurs Pratiques qui contient les données sur les pratiques
`static java.util.Vector`	`ListeProcessus` Vecteur des Vecteurs Processus qui contient les données sur

	les processus (enregistrements du fichier processus.xml)
static java.util.Vector	ListeRessources Vecteur des Vecteurs Ressources (enregistrements du fichier ressource.xml)
static java.util.Vector	ListeTermes Vecteur des Vecteurs Termes du Glossaire (enregistrements du fichier termeglossaire.xml)
static javax.swing.JMenuItem	menuFitemEH
static javax.swing.JMenuItem	menuFitemEX
static javax.swing.JMenuItem	menuFitemFS
static javax.swing.JMenuItem	menuFitemG
static javax.swing.JMenuItem	menuFitemMO
static javax.swing.JMenuItem	menuFitemQ
static javax.swing.JMenuItem	menuFitemQS
static boolean	Modifie Booléen de modification de la bas des données
static Primitives	Prim Ocurence de Primitives
static java.util.Vector	RessourcesUtilisees Vecteur des Vecteurs Ressources Utilisés (enregistrements du fichier utilisation.xml)
static boolean	SauveXml Booléen de sauvegarde des fichiers xml
static java.util.Vector	SavedEntrees Vecteur des Vecteurs Entrées sauvegardés (enregistrements du fichier entree.xml)
static java.util.Vector	SavedListeDelivrables Vecteur des Vecteurs Delivrables sauvegardées (enregistrements du fichier delivrable.xml)
static java.util.Vector	SavedListePratiques Vecteur des Vecteurs qu'on sauve Pour récuperer si on veut annuler les opératiosn effectuées
static java.util.Vector	SavedListeRessources Vecteur des Vecteurs Ressources sauvegardés(enregistrements du fichier ressource.xml)
static java.util.Vector	SavedListeTermes Vecteur des Vecteurs Termes du Glossaire sauvegardés (enregistrements du fichier termeglossaire.xml)

static java.util.Vector	SavedRessourcesUtilisees Vecteur des Vecteurs Ressources Utilisées saugardés (enregistrements du fichier utilisation.xml)
static java.util.Vector	SavedSorties Vecteur des Vecteurs Sorties sauvegardés (enregistrements du fichier sortie.xml)
static java.util.Vector	Sorties Vecteur des Vecteurs Sorties (enregistrements du fichier sortie.xml)

Fields inherited from class javax.swing.JFrame
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

OM()

Method Summary

protected void	createFrame()
protected void	createFrameAccueil()
protected void	createFrameProcessus()
protected void	createFrameTGlossaire()
protected javax.swing.JMenuBar	createMenuBar()
static void	main(java.lang.String[] args)

Methods inherited from class javax.swing.JFrame
addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isRootPaneCheckingEnabled, paramString, processKeyEvent,

processWindowEvent, setContentPane, setDefaultCloseOperation, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame

addNotify, finalize, getCursorType, getFrames, getIconImage, getMenuBar, getState, getTitle, isResizable, remove, removeNotify, setCursor, setIconImage, setMenuBar, setResizable, setState, setTitle

Methods inherited from class java.awt.Window

addWindowListener, applyResourceBundle, applyResourceBundle, dispose, getFocusOwner, getInputContext, getLocale, getOwnedWindows, getOwner, getToolkit, getWarningString, isShowing, pack, postEvent, processEvent, removeWindowListener, show, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, remove, removeAll, removeContainerListener, setFont, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isFocusTraversable, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processInputMethodEvent, processMouseEvent, processMouseMotionEvent, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

desktop

```
public static javax.swing.JDesktopPane desktop
```

---

menuFitemMO

```
public static javax.swing.JMenuItem menuFitemMO
```

---

menuFitemG

```
public static javax.swing.JMenuItem menuFitemG
```

---

menuFitemFS

```
public static javax.swing.JMenuItem menuFitemFS
```

---

menuFitemQS

```
public static javax.swing.JMenuItem menuFitemQS
```

---

menuFitemEX

```
public static javax.swing.JMenuItem menuFitemEX
```

---

menuFitemEH

```
public static javax.swing.JMenuItem menuFitemEH
```

---

menuFitemQ

```
public static javax.swing.JMenuItem menuFitemQ
```

Prim

```
public static Primitives Prim
```

Ocurence de Primitives

See Also:

Primitives

ListeProcessus

```
public static java.util.Vector ListeProcessus
```

Vecteur des Vecteurs Processus qui contient les données sur les processus (enregistrements du fichier processus.xml)

ListeNomProcessus

```
public static java.util.Vector ListeNomProcessus
```

Vecteur qui contient les noms des processus pour faire le choix

---

ListePratiques

```
public static java.util.Vector ListePratiques
```

Vecteur des Vecteurs Pratiques qui contient les données sur les pratiques

---

SavedListePratiques

```
public static java.util.Vector SavedListePratiques
```

Vecteur des Vecteurs qu'on sauve Pour récuperer si on veut annuler les opératiosn effectuées

---

Entrees

```
public static java.util.Vector Entrees
```

Vecteur des Vecteurs Entrées (enregistrements du fichier entree.xml)

SavedEntrees

```
public static java.util.Vector SavedEntrees
```

Vecteur des Vecteurs Entrées sauvegardés (enregistrements du fichier entree.xml)

Sorties

```
public static java.util.Vector Sorties
```

Vecteur des Vecteurs Sorties (enregistrements du fichier sortie.xml)

SavedSorties

```
public static java.util.Vector SavedSorties
```

Vecteur des Vecteurs Sorties sauvegardés (enregistrements du fichier sortie.xml)

RessourcesUtilisees

```
public static java.util.Vector RessourcesUtilisees
```

Vecteur des Vecteurs Ressources Utilisés (enregistrements du fichier utilisation.xml)

SavedRessourcesUtilisees

```
public static java.util.Vector SavedRessourcesUtilisees
```

Vecteur des Vecteurs Ressources Utilisées saugardés (enregistrements du fichier utilisation.xml)

ListeDelivrables

```
public static java.util.Vector ListeDelivrables
```

Vecteur des Vecteurs Delivrables (enregistrements du fichier delivrable.xml)

ListeRessources

```
public static java.util.Vector ListeRessources
```

Vecteur des Vecteurs Ressources (enregistrements du fichier ressource.xml)

SavedListeDelivrables

```
public static java.util.Vector SavedListeDelivrables
```

Vecteur des Vecteurs Delivrables sauvegardées (enregistrements du fichier delivrable.xml)

SavedListeRessources

```
public static java.util.Vector SavedListeRessources
```

Vecteur des Vecteurs Ressources sauvegardés(enregistrements du fichier ressource.xml)

ListeTermes

```
public static java.util.Vector ListeTermes
```

Vecteur des Vecteurs Termes du Glossaire (enregistrements du fichier termeglossaire.xml)

SavedListeTermes

```
public static java.util.Vector SavedListeTermes
```

Vecteur des Vecteurs Termes du Glossaire sauvegardés (enregistrements du fichier termeglossaire.xml)

Modifie

```
public static boolean Modifie
```

Booléen de modification de la bas des données

SauveXml

```
public static boolean SauveXml
```

Booléen de sauvegarde des fichiers xml

Constructor Detail

OM

```
public OM()
```

Method Detail

createMenuBar

```
protected javax.swing.JMenuBar createMenuBar()
```

---

createFrame

```
protected void createFrame()
```

---

createFrameAccueil

```
protected void createFrameAccueil()
```

---

createFrameProcessus

```
protected void createFrameProcessus()
```

---

createFrameTGlossaire

```
protected void createFrameTGlossaire()
```

---

main

```
public static void main(java.lang.String[] args)
```

Class Primitives

```
java.lang.Object
  |
  +--Primitives
```

---

public class **Primitives**
extends java.lang.Object

Fichier : Primitives.java

Classe : Classe Primitives.Les méthodes de cette classe sont des primitives qu'on utilise plus d'une fois. Toutes les autres classes doivent l'instancier, avant d'utiliser ses méthodes.

Note : Certaines librairies spéciales utilisées ici doivent être téléchargées car elles ne font pas partie de java 2 standard. Il s'agit de **javax.xml.org.w3c.com** (http://www.java.sun.com/xml) et de **com.sun.xml.parser** (htp://www.java.sun.com/xml) le parseur ProjectX de chez Sun.

Note importante: Les vecteurs formés dans les méthodes de cette classe (après parsing des fichiers xml) contiennent des vecteurs qui sont chacun un vecteur des strings qui contient tous les éléments (au sens xml) de l'élément avec la balise principale du fichier.

Par exemple si on parse le fichier composé des éléments avec la balise "exemple" composé chacun de 2 éléments avc comme balises "el1" et "el2"
On a un vecteur des vecteurs composés de 2 strings.
Il faut veiller à faire attention au fait que le premier élément d'un vecteur est d'indice 0.

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)

Constructor Summary

`Primitives()`

Method Summary

boolean	`appartient(java.util.Vector Liste, java.lang.String Donnee, int Position)` Est à vrai s'il existe dans le vecteur Liste, un vecteur qui a la valeur du String Donnee à la position Position
void	`ArrangePratiques(java.util.Vector VecIndexes, java.util.Vector ListePratiques, java.lang.String RefProcessus, java.util.Vector VecESR)`

	Cette fonction arrange la référence des pratiques d'un processus après un changement de leur nombre (ajout ou suppression).
void	`CopieVecteur(java.util.Vector Source, java.util.Vector Cible)` Copie le contenu du vecteur Source qur le vecteur Cible
void	`DeleteAtVector(java.util.Vector Donne, int Position, java.lang.String Valeur)` Supprime dans le vecteur Donne le vecteur qui, à la position Position, a la valeur du String Valeur.
static void	`ListeAlpha(java.util.Vector Donnees, java.util.Vector VecOut)` Met dans VecOut les mots dont les premières lettres sont les lettres de l'alphabet présentent dans le Glossaire (repréenté par le vectuer Donnees)
void	`Parser(java.lang.String Filename, java.util.Vector VecteurResultat, java.lang.String balise)` Parse le fichier "Filename".
void	`RemplirNoms(java.util.Vector VecteurIn, java.util.Vector VecteurOut, int Position)` RemplirNoms prend tous les strings à la position "Position" du vecteur des vecteurs "VecteurIn" et forme un vecteur de ces strings en respectant leur ordre dans le vecteur VecteurIn.
static void	`SauverFichierHtml()` Sauvegarde des fichiers html du modèle OWPL
static void	`SauverFichierXml()` Sauvegarde de tous les fichiers xml
static void	`SauverHTMLProcessus(java.util.Vector Donnees)` Sauvegarde le fichier ModeleOWPL.html à partir du vecteur Donnees
static void	`SauverHTMLProcessusPratiques(int Numero, java.util.Vector ProcessusA)` Sauvegarde le fichier ProcessusXXX.html à partir du vecteur ProcessusA avec toutes ses pratiques.
static void	`SauverHTMLTermeglossaire(java.util.Vector Donnees)` Sauvegarde le fichier Glossaire.html à partir du vecteur Donnees
static void	`SauverTableDelivrable(java.util.Vector Donnees)` Sauvegarde le fichier delivrable.xml à partir du vecteur Donnees
static void	`SauverTableEntree(java.util.Vector Donnees)` Sauvegarde le fichier entree.xml à partir du vecteur Donnees
static void	`SauverTablePratique(java.util.Vector Donnees)` Sauvegarde le fichier pratique.xml à partir du vecteur Donnees
static void	`SauverTableProcessus(java.util.Vector Donnees)` Sauvegarde le fichier processus.xml à partir du vecteur Donnees
static void	`SauverTableRessource(java.util.Vector Donnees)` Sauvegarde le fichier ressource.xml à partir du vecteur Donnees
static void	`SauverTableSortie(java.util.Vector Donnees)` Sauvegarde le fichier sortie.xml à partir du vecteur Donnees
static void	`SauverTableTermeGlossaire(java.util.Vector Donnees)` Sauvegarde le fichier termeglossaire.xml à partir du vecteur Donnees
static void	`SauverTableUtilisation(java.util.Vector Donnees)` Sauvegarde le fichier utilisation.xml à partir du vecteur Donnees

void	`SelectIndexes(java.util.Vector VecteurDonne, int Pos, java.util.Vector VecteurRes, java.lang.String Valeur)`
static void	`SelectWhere(java.util.Vector VecteurDonne, int Pos, java.util.Vector VecteurRes, java.lang.String Valeur)` SelectWhere donne dans VecteurRes, tous les vecteurs contenus dans VecteurDonne qui ont à la position Pos, un string qui est égal à "Valeur".
static void	`SuppressionProcessus(java.lang.String RefPro)` Suppression du processus avec la référence RefPro dans le système avec suppression en cascade de ses pratiques, et des lines que ses pratiques ont avec les délivrables et les ressources.
static void	`SupprimerCouple(java.util.Vector Couple, java.util.Vector Liste)` Suppression du vecteur ayant les mêmes valeurs que le vecteur Couple dans le vecteur Liste
void	`UpdatePratiquesApresAjout(java.util.Vector VPra, java.util.Vector IPra, int Pos, java.util.Vector Pra, java.util.Vector VESR)` Après ajout d'une nouvelle pratique à un processus, cette fonction ajoute la nouvelle pratique à l'endoit voulu par l'utilisateur et renomme les références de l'ensemble des pratiques du processus VPra : Vecteur des vecteurs contenant toutes les pratiques IPra : Vecteur contenant les indexes des pratiques du processus édité Pos : Position où on veut inserer la nouvelle pratique Pra : Nouvelle pratique à ajouter VESR : Vecteur conteant les listes des entres, sorties et ressources du système
void	`UpdateReference(java.util.Vector VDonne, java.lang.String OldRef, java.lang.String NewRef, int Pos)` Update change toutes les références de pratiques dans la vecteur VDonne ayant comme valeur de référence OldRef en la valeur NewRef.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Primitives

```
public Primitives()
```

Method Detail

Parser

```
public void Parser(java.lang.String Filename,
                   java.util.Vector VecteurResultat,
                   java.lang.String balise)
```

Parse le fichier "Filename". Les éléments avec la balise "balise" sont pris pour former le vecteur "VecteurResultat".

Cette méthode est utilisée quand on veut ouvrir les fichiers xml au lancemnet de l'application OWPL Manager.

RemplirNoms

```
public void RemplirNoms(java.util.Vector VecteurIn,
                        java.util.Vector VecteurOut,
                        int Position)
```

RemplirNoms prend tous les strings à la position "Position" du vecteur des vecteurs "VecteurIn" et forme un vecteur de ces strings en respectant leur ordre dans le vecteur VecteurIn.

SelectWhere

```
public static void SelectWhere(java.util.Vector VecteurDonne,
                               int Pos,
                               java.util.Vector VecteurRes,
                               java.lang.String Valeur)
```

SelectWhere donne dans VecteurRes, tous les vecteurs contenus dans VecteurDonne qui ont à la position Pos, un string qui est égal à "Valeur".

SelectIndexes

```
public void SelectIndexes(java.util.Vector VecteurDonne,
                          int Pos,
                          java.util.Vector VecteurRes,
                          java.lang.String Valeur)
```

DeleteAtVector

```
public void DeleteAtVector(java.util.Vector Donne,
                           int Position,
                           java.lang.String Valeur)
```

Supprime dans le vecteur Donne le vecteur qui, à la position Position, a la valeur du String Valeur.

Spécialemnt écrit pour les pratiques, cette fonction supprime en cascade les entrées, sorties et ressources qui sont liés à une pratique.
Elle est donc utilisée quand on veut supprimer une pratique d'un processus.

ArrangePratiques

```
public void ArrangePratiques(java.util.Vector VecIndexes,
                             java.util.Vector ListePratiques,
                             java.lang.String RefProcessus,
                             java.util.Vector VecESR)
```

Cette fonction arrange la référence des pratiques d'un processus après un changement de leur nombre (ajout ou suppression).
VecIndexes contient les index des pratiques du processus dans le vecteur ListePratiques des pratiques du système. RefProcessus contient la référence du processus.
Pour garder la cohérence dans le système, pour chaque référence de pratique changée, il faut également changer les clés étrangères dans les Entrées, les Sorties et les Ressources
VecESR contient les Vecteurs des Entrées, Sorties et Ressources du système

---

UpdateReference

```
public void UpdateReference(java.util.Vector VDonne,
                            java.lang.String OldRef,
                            java.lang.String NewRef,
                            int Pos)
```

Update change toutes les références de pratiques dans la vecteur VDonne ayant comme valeur de référence OldRef en la valeur NewRef.
Ces references dans les vecteurs de VDonne sont à la position Pos

---

UpdatePratiquesApresAjout

```
public void UpdatePratiquesApresAjout(java.util.Vector VPra,
                                      java.util.Vector IPra,
                                      int Pos,
                                      java.util.Vector Pra,
                                      java.util.Vector VESR)
```

Après ajout d'une nouvelle pratique à un processus, cette fonction ajoute la nouvelle pratique à l'endoit voulu par l'utilisateur et renomme les références de l'ensemble des pratiques du processus
VPra : Vecteur des vecteurs contenant toutes les pratiques
IPra : Vecteur contenant les indexes des pratiques du processus édité
Pos : Position où on veut inserer la nouvelle pratique
Pra : Nouvelle pratique à ajouter
VESR : Vecteur conteant les listes des entres, sorties et ressources du système

---

CopieVecteur

```
public void CopieVecteur(java.util.Vector Source,
                         java.util.Vector Cible)
```

Copie le contenu du vecteur Source qur le vecteur Cible

SauverTableEntree

```
public static void SauverTableEntree(java.util.Vector Donnees)
```

Sauvegarde le fichier entree.xml à partir du vecteur Donnees

SauverTableSortie

```
public static void SauverTableSortie(java.util.Vector Donnees)
```

Sauvegarde le fichier sortie.xml à partir du vecteur Donnees

SauverTableUtilisation

```
public static void SauverTableUtilisation(java.util.Vector Donnees)
```

Sauvegarde le fichier utilisation.xml à partir du vecteur Donnees

---

SauverTableProcessus

```
public static void SauverTableProcessus(java.util.Vector Donnees)
```

Sauvegarde le fichier processus.xml à partir du vecteur Donnees

---

SauverTablePratique

```
public static void SauverTablePratique(java.util.Vector Donnees)
```

Sauvegarde le fichier pratique.xml à partir du vecteur Donnees

---

SauverTableDelivrable

```
public static void SauverTableDelivrable(java.util.Vector Donnees)
```

Sauvegarde le fichier delivrable.xml à partir du vecteur Donnees

---

SauverTableRessource

```
public static void SauverTableRessource(java.util.Vector Donnees)
```

Sauvegarde le fichier ressource.xml à partir du vecteur Donnees

---

SauverTableTermeGlossaire

```
public static void SauverTableTermeGlossaire(java.util.Vector Donnees)
```

Sauvegarde le fichier termeglossaire.xml à partir du vecteur Donnees

---

SauverFichierXml

```
public static void SauverFichierXml()
```

Sauvegarde de tous les fichiers xml

SauverHTMLProcessus

```
public static void SauverHTMLProcessus(java.util.Vector Donnees)
```

Sauvegarde le fichier ModeleOWPL.html à partir du vecteur Donnees

---

SauverHTMLProcessusPratiques

```
public static void SauverHTMLProcessusPratiques(int Numero,
                                                java.util.Vector ProcessusA)
```

Sauvegarde le fichier ProcessusXXX.html à partir du vecteur ProcessusA avec toutes ses pratiques.
Numero est le numéro du processus dans le modèle OWPL

ListeAlpha

```
public static void ListeAlpha(java.util.Vector Donnees,
                              java.util.Vector VecOut)
```

Met dans VecOut les mots dont les premières lettres sont les lettres de l'alphabet présentent dans le Glossaire (repréenté par le vectuer Donnees)

SauverHTMLTermeglossaire

```
public static void SauverHTMLTermeglossaire(java.util.Vector Donnees)
```

Sauvegarde le fichier Glossaire.html à partir du vecteur Donnees

SauverFichierHtml

```
public static void SauverFichierHtml()
```

Sauvegarde des fichiers html du modèle OWPL

SupprimerCouple

```
public static void SupprimerCouple(java.util.Vector Couple,
                                   java.util.Vector Liste)
```

Suppression du vecteur ayant les mêmes valeurs que le vecteur Couple dans le vecteur Liste

---

SuppressionProcessus

```
public static void SuppressionProcessus(java.lang.String RefPro)
```

Suppression du processus avec la référence RefPro dans le système avec suppression en cascade de ses pratiques, et des lines que ses pratiques ont avec les délivrables et les ressources.

---

appartient

```
public boolean appartient(java.util.Vector Liste,
                          java.lang.String Donnee,
                          int Position)
```

Est à vrai s'il existe dans le vecteur Liste, un vecteur qui a la valeur du String Donnee à la position Position

---

Class Processus

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--Processus
```

---

public class **Processus**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener

Fichier : Processus.java

Classe : Fenêtre processus qui manipule les processus dans le modèle OWPL. Cette fenêtre fait partie de l'application OWPL Manager

Cette fenêtre commence par lire tous les fichiers xml du système puis forme les différents vecteurs des enregistrements. Elle permet ensuite d'acceder au **glossaire**, aux **questionnaires**, aux **facteurs de succès** et aux **processus** du modèle OWPL. L'accès aux processus du modèle est possible via une Liste.

Quand une fichier xml est lu, chaque enregistrement est representé comme un vectuer avec autant de vectuers qu'il y a des champs (par exemple, un enregistrement du fichier entree.xml est un Vecteuer comportant deux strings: le nom du délivrable et le référence de la pratique). Ces vectuers sont ensuite assemblés pour former un seul vecteur des vectuers qui rerpésenta ainsi l'ensemble des enregistrements. (concernant l'exemple sur entree.xml il s'agit du vecteur Entrees)

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Field Summary	
static javax.swing.JDesktopPane	desktop
static javax.swing.JButton	Editer
static javax.swing.JButton	EnregistrerHTML
static javax.swing.JButton	EnregistrerXML
static java.util.Vector	Entrees Vecteur des Vecteurs Entrées (enregistrements du fichier entree.xml)
static javax.swing.JList	ListeChoix Liste des processus
static java.util.Vector	ListeDelivrables Vecteur des Vecteurs Delivrables (enregistrements du fichier delivrable.xml)
static java.util.Vector	ListeNomProcessus Vecteur qui contient les noms des processus pour faire le choix
static java.util.Vector	ListePratiques Vecteur des Vecteurs Pratiques qui contient les données sur les pratiques
static java.util.Vector	ListeProcessus Vecteur des Vecteurs Processus qui contient les données sur les processus (enregistrements du fichier processus.xml)
static java.util.Vector	ListeRessources Vecteur des Vecteurs Ressources (enregistrements du fichier ressource.xml)
static java.util.Vector	ListeTermes Vecteur des Vecteurs Termes du Glossaire (enregistrements du fichier termeglossaire.xml)
static boolean	Modifie
static javax.swing.JPanel	p2
static javax.swing.JPanel	p3CenterNorth
static java.util.Vector	RessourcesUtilisees Vecteur des Vecteurs Ressources Utilisés (enregistrements du fichier utilisation.xml)
static boolean	SauveXML
static java.util.Vector	SavedEntrees

	Vecteur des Vecteurs Entrées sauvegardés (enregistrements du fichier entree.xml)
static java.util.Vector	SavedListeDelivrables Vecteur des Vecteurs Delivrables sauvegardées (enregistrements du fichier delivrable.xml)
static java.util.Vector	SavedListePratiques Vecteur des Vecteurs qu'on sauve Pour récuperer si on veut annuler les opératiosn effectuées
static java.util.Vector	SavedListeRessources Vecteur des Vecteurs Ressources sauvegardés(enregistrements du fichier ressource.xml)
static java.util.Vector	SavedListeTermes Vecteur des Vecteurs Termes du Glossaire sauvegardés (enregistrements du fichier termeglossaire.xml)
static java.util.Vector	SavedRessourcesUtilisees Vecteur des Vecteurs Ressources Utilisées saugardés (enregistrements du fichier utilisation.xml)
static java.util.Vector	SavedSorties Vecteur des Vecteurs Sorties sauvegardés (enregistrements du fichier sortie.xml)
static javax.swing.JScrollPane	ScrolListe Conteneur de la liste des processus
static java.util.Vector	Sorties Vecteur des Vecteurs Sorties (enregistrements du fichier sortie.xml)
static javax.swing.JButton	Supprimer
static TermeGlossaire	TG
static javax.swing.JButton	Visualiser

Fields inherited from class javax.swing.JInternalFrame
closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

Processus()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)

Methods inherited from class javax.swing.JInternalFrame

addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable, isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected, setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize,

paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

Supprimer

public static javax.swing.JButton **Supprimer**

Editer

public static javax.swing.JButton **Editer**

Visualiser

public static javax.swing.JButton **Visualiser**

EnregistrerXML

public static javax.swing.JButton **EnregistrerXML**

EnregistrerHTML

```
public static javax.swing.JButton EnregistrerHTML
```

ListeChoix

```
public static javax.swing.JList ListeChoix
```

Liste des processus

ScrolListe

```
public static javax.swing.JScrollPane ScrolListe
```

Conteneur de la liste des processus

ListeProcessus

```
public static java.util.Vector ListeProcessus
```

Vecteur des Vecteurs Processus qui contient les données sur les processus (enregistrements du fichier processus.xml)

ListeNomProcessus

```
public static java.util.Vector ListeNomProcessus
```

Vecteur qui contient les noms des processus pour faire le choix

---

ListePratiques

```
public static java.util.Vector ListePratiques
```

Vecteur des Vecteurs Pratiques qui contient les données sur les pratiques

---

SavedListePratiques

```
public static java.util.Vector SavedListePratiques
```

Vecteur des Vecteurs qu'on sauve Pour récuperer si on veut annuler les opératiosn effectuées

---

Entrees

```
public static java.util.Vector Entrees
```

Vecteur des Vecteurs Entrées (enregistrements du fichier entree.xml)

SavedEntrees

```
public static java.util.Vector SavedEntrees
```

Vecteur des Vecteurs Entrées sauvegardés (enregistrements du fichier entree.xml)

Sorties

```
public static java.util.Vector Sorties
```

Vecteur des Vecteurs Sorties (enregistrements du fichier sortie.xml)

SavedSorties

```
public static java.util.Vector SavedSorties
```

Vecteur des Vecteurs Sorties sauvegardés (enregistrements du fichier sortie.xml)

RessourcesUtilisees

```
public static java.util.Vector RessourcesUtilisees
```

Vecteur des Vecteurs Ressources Utilisés (enregistrements du fichier utilisation.xml)

SavedRessourcesUtilisees

```
public static java.util.Vector SavedRessourcesUtilisees
```

Vecteur des Vecteurs Ressources Utilisées saugardés (enregistrements du fichier utilisation.xml)

ListeDelivrables

```
public static java.util.Vector ListeDelivrables
```

Vecteur des Vecteurs Delivrables (enregistrements du fichier delivrable.xml)

ListeRessources

```
public static java.util.Vector ListeRessources
```

Vecteur des Vecteurs Ressources (enregistrements du fichier ressource.xml)

SavedListeDelivrables

```
public static java.util.Vector SavedListeDelivrables
```

Vecteur des Vecteurs Delivrables sauvegardées (enregistrements du fichier delivrable.xml)

SavedListeRessources

```
public static java.util.Vector SavedListeRessources
```

Vecteur des Vecteurs Ressources sauvegardés(enregistrements du fichier ressource.xml)

ListeTermes

```
public static java.util.Vector ListeTermes
```

Vecteur des Vecteurs Termes du Glossaire (enregistrements du fichier termeglossaire.xml)

SavedListeTermes

```
public static java.util.Vector SavedListeTermes
```

Vecteur des Vecteurs Termes du Glossaire sauvegardés (enregistrements du fichier termeglossaire.xml)

---

p2

```
public static javax.swing.JPanel p2
```

---

p3CenterNorth

```
public static javax.swing.JPanel p3CenterNorth
```

Modifie

```
public static boolean Modifie
```

SauveXML

```
public static boolean SauveXML
```

TG

```
public static TermeGlossaire TG
```

---

desktop

```
public static javax.swing.JDesktopPane desktop
```

Constructor Detail

Processus

```
public Processus()
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

Class TermeGlossaire

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JInternalFrame
                          |
                          +--TermeGlossaire
```

public class **TermeGlossaire**
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener

Fichier : TermeGlossaire.java

Classe : Fenêtre TermeGlossaire qui manipule les terme du glossaire dans le modèle OWPL. Cette fenêtre fait partie de l'application OWPL Manager

Author:
Bobo MAKUNDA SEFEKESE (Avril 2001)
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JInternalFrame
javax.swing.JInternalFrame.AccessibleJInternalFrame, javax.swing.JInternalFrame.JDesktopIcon

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Field Summary

static javax.swing.JTextArea	Contenu
static javax.swing.JButton	Editer
static javax.swing.JComboBox	ListeChoix Liste déroulante de choix de terme

static java.util.Vector	ListeIntitules Vecteur qui contient les intitules pour faire le choix
static java.util.Vector	ListeTermes Vecteur des Vecteurs Terme qui contient les données sur les termes
static javax.swing.JPanel	p3CenterNorth panel qui contient la liste des termes
static javax.swing.JButton	Supprimer

Fields inherited from class javax.swing.JInternalFrame

closable, CONTENT_PANE_PROPERTY, desktopIcon, FRAME_ICON_PROPERTY, frameIcon, GLASS_PANE_PROPERTY, iconable, IS_CLOSED_PROPERTY, IS_ICON_PROPERTY, IS_MAXIMUM_PROPERTY, IS_SELECTED_PROPERTY, isClosed, isIcon, isMaximum, isSelected, LAYERED_PANE_PROPERTY, maximizable, MENU_BAR_PROPERTY, resizable, ROOT_PANE_PROPERTY, rootPane, rootPaneCheckingEnabled, title, TITLE_PROPERTY

Fields inherited from class javax.swing.JComponent

accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary

TermeGlossaire()

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
static void	Refresh(javax.swing.JTextArea Espace, java.util.Vector Donnees)

Methods inherited from class javax.swing.JInternalFrame

addImpl, addInternalFrameListener, createRootPane, dispose, fireInternalFrameEvent, getAccessibleContext, getBackground, getContentPane, getDefaultCloseOperation, getDesktopIcon, getDesktopPane, getForeground, getFrameIcon, getGlassPane, getJMenuBar, getLayer, getLayeredPane, getMenuBar, getRootPane, getTitle, getUI, getUIClassID, getWarningString, isClosable, isClosed, isIcon, isIconifiable, isMaximizable, isMaximum, isResizable, isRootPaneCheckingEnabled, isSelected, moveToBack, moveToFront, pack, paramString, removeInternalFrameListener, reshape, setBackground, setClosable, setClosed, setContentPane, setDefaultCloseOperation, setDesktopIcon, setForeground, setFrameIcon, setGlassPane, setIcon, setIconifiable, setJMenuBar, setLayer, setLayeredPane, setLayout, setMaximizable, setMaximum, setMenuBar, setResizable, setRootPane, setRootPaneCheckingEnabled, setSelected,

setTitle, setUI, setVisible, show, toBack, toFront, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInsets, getInsets, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, processComponentKeyEvent, processFocusEvent, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, revalidate, scrollRectToVisible, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hide, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, size, toString, transferFocus

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

Supprimer

public static javax.swing.JButton **Supprimer**

Editer

public static javax.swing.JButton **Editer**

ListeChoix

public static javax.swing.JComboBox **ListeChoix**

Liste déroulante de choix de terme

Contenu

public static javax.swing.JTextArea **Contenu**

ListeTermes

public static java.util.Vector **ListeTermes**

Vecteur des Vecteurs Terme qui contient les données sur les termes

---

ListeIntitules

public static java.util.Vector **ListeIntitules**

Vecteur qui contient les intitules pour faire le choix

---

p3CenterNorth

public static javax.swing.JPanel **p3CenterNorth**

panel qui contient la liste des termes

Constructor Detail

TermeGlossaire

```
public TermeGlossaire()
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
actionPerformed in interface java.awt.event.ActionListener

Refresh

```
public static void Refresh(javax.swing.JTextArea Espace,
                           java.util.Vector Donnees)
```

VII. INDEX

INDEX

O

P

Q

R

S

T

X